Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αριθμομηχανή/Μετατροπέας αριθμητικών συστημάτων – Εφαρμογή σε Android

Μουστάκας Νικόλαος ΑΜ 39096
Δημητρίου Ορέστης ΑΜ 39102

Εισηγητής: Ματιάτος Σπυρίδων, Λέκτορας Εφαρμογών

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης:

Εισαγωγή

Η παρούσα πτυχιακή εργασία ασχολείται με τους αριθμούς και την αναπαράσταση τους στα πιο διαδεδομένα αριθμητικά συστήματα πέραν του δεκαδικού που χρησιμοποιούμε στη καθημερινότητά μας.

Όχι τόσο η πολυπλοκότητα αλλά κυρίως η ταχύτητα που επιθυμεί ο καθένας μας για να πραγματοποιήσει αριθμητικές πράξεις, σε ένα αριθμητικό σύστημα όπως είναι το δυαδικό ή ακόμα και μεταξύ αριθμών που ανήκουν σε διαφορετικά αριθμητικά συστήματα, αποτέλεσε το σκοπό της δημιουργίας μίας εφαρμογής για έξυπνα κινητά, η οποία θα ενσωμάτωνε ταυτοχρόνως δύο ισχυρά εργαλεία. Πρώτον έναν μετατροπέα και δεύτερον μία αριθμομηχανή.

Πιο συγκεκριμένα η εφαρμογή αναπτύχθηκε έτσι ώστε να εγκατασταθεί σε συσκευές που χρησιμοποιούν ως λειτουργικό σύστημα το Android, ένα λειτουργικό σύστημα που κατέχει ένα μερίδιο άνω του 87% στη παγκόσμια αγορά λειτουργικού συστήματος για φορητές συσκευές.

Η εφαρμογή θα είναι διαθέσιμη προς εγκατάσταση μέσω του Google Play, μία υπηρεσία ψηφιακής διανομής, διάφορων μέσων όπως εφαρμογές, ταινίες και τραγούδια που λειτουργεί και αναπτύσσεται από την ίδια τη Google.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1ο :

Αριθμητικά Συστήματα - Μετατροπές – Πράξεις - Συμπληρώματα ως προς 1 και 2

1.1 Αριθμητικά συστήματα – Εισαγωγή

Ένα αριθμητικό σύστημα ορίζεται από ένα σύνολο τιμών, που χρησιμοποιούνται για την αναπαράσταση μίας ποσότητας. Η μελέτη των αριθμητικών συστημάτων δεν περιορίζεται μόνο στους ηλεκτρονικούς υπολογιστές. Κάθε μέρα χρησιμοποιούμε αριθμούς, και γνωρίζοντας τον τρόπο λειτουργίας τους, μας παρέχεται η πληροφορία για το πώς ένας υπολογιστής χειρίζεται και αποθηκεύει τους αριθμούς. Η ανθρωπότητα, κατά τη διάρκεια της πορείας της, χρησιμοποίησε σχήματα ή σύμβολα για την αναπαράσταση των αριθμών, αρχικά με τη χρήση γραμμών. Η γραφική απεικόνιση αριθμών ήταν όμως δύσκολη. Οι Βαβυλώνιοι χρησιμοποιούσαν το εξηνταδικό σύστημα (sexagesimal) οι δε Μάγια το εικοσαδικό (vigesimal). Οι Ρωμαίοι επινόησαν ένα αριθμητικό σύστημα για τους αριθμούς από το 1 έως το 1 εκατομμύριο με τη χρήση 7 συμβόλων (τα γράμματα I, V, X, L, C, D, M). Το γνωστό μας αραβικό σύστημα χρησιμοποιήθηκε για πρώτη φορά πριν μόλις 2300 χρόνια. Η εισαγωγή του μηδενός, ώστε να προσδιορίσει την αξία ενός ψηφίου, ήταν πολύ σημαντική. (Maxfield, 2009; Αριθμητικά Συστήματα, 2015) Οι ηλεκτρονικοί υπολογιστές εκτελούν τις πράξεις, χρησιμοποιώντας όχι το δεκαδικό σύστημα αρίθμησης αλλά το δυαδικό, που περιλαμβάνει τους δυαδικούς αριθμούς «1» και «0». Τα ψηφιακά ηλεκτρονικά είναι ο κόσμος των υπολογιστών, των ολοκληρωμένων κυκλωμάτων, των αριθμομηχανών. Η χρήση των ψηφιακών ηλεκτρονικών επεκτείνεται σε πολλαπλές εφαρμογές της καθημερινής μας ζωής και, για αυτό το λόγο, θα πρέπει να γνωρίζουμε τις βασικές αρχές τους.

1.2. Κύρια αριθμητικά συστήματα

Το πλέον χρησιμοποιούμενο σύστημα αρίθμησης, στην καθημερινή μας ζωή, είναι το δεκαδικό. Αυτό περιλαμβάνει τα ψηφία από 0 έως 9 και έχει σαν βάση του (base, radix) τον αριθμό 10. Παρατηρούμε ότι η βάση του συστήματος δεν ανήκει στα ψηφία του συστήματος, αλλά υπερβαίνει κατά μία μονάδα το μεγαλύτερο ψηφίο του. Το ίδιο ισχύει για όλα τα αριθμητικά συστήματα, ανεξάρτητα από τη βάση, σύμφωνα με την οποία αυτά έχουν ορισθεί.

Εκτός από το δεκαδικό αριθμητικό σύστημα, τα κύρια αριθμητικά συστήματα με τα οποία θα ασχοληθούμε είναι το δυαδικό, το οκταδικό και το δεκαεξαδικό. Όπως αναφέραμε, το δυαδικό σύστημα είναι αυτό που χρησιμοποιείται στους ηλεκτρονικούς υπολογιστές. Τα δύο ψηφία, που το περιγράφουν, μπορούν να παρασταθούν εύκολα με τη βοήθεια ηλεκτρονικών εξαρτημάτων, όπως οι διακόπτες ή τα τρανζίστορ, των οποίων οι έξοδοι μπορούν να οδηγηθούν σε δύο καταστάσεις. Η μία από αυτές παριστάνει το «0» και η άλλη το «1» (με τη χρήση διακοπτών η πρώτη είναι ανοικτός διακόπτης και η δεύτερη κλειστός διακόπτης). Τα ψηφία που χρησιμοποιεί το κάθε σύστημα αρίθμησης είναι (για τα κύρια συστήματα):

ΔΥΑΔΙΚΟ: 0, 1

ΟΚΤΑΔΙΚΟ: 0, 1, 2, 3, 4, 5, 6, 7

ΔΕΚΑΔΙΚΟ: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

ΔΕΚΑΕΞΑΔΙΚΟ : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

1.3 Μετατροπή ακέραιων αριθμών από το δεκαδικό σύστημα

Βασική αρχή είναι ότι γράφουμε τον αριθμό μας δεξιά και κατόπιν εκτελούμε συνεχείς διαιρέσεις με τη βάση του συστήματος, στο οποίο θέλουμε να μετατρέψουμε τον αριθμό μας. Στο ίδιο επίπεδο με τον αρχικό αριθμό γράφουμε το ακέραιο πηλίκο (DIV) και ακριβώς από κάτω το ακέραιο υπόλοιπο(MOD). Σταματάμε τη διαδικασία όταν το πηλίκο γίνει μικρότερο από τον αριθμό με τον οποίο διαιρούμε, αποτέλεσμα το οποίο κατεβάζουμε στη θέση του υπολοίπου. Η ανάγνωση του αριθμού που προκύπτει, από αριστερά προς τα δεξιά, δίνει τον αριθμό στο νέο αριθμητικό σύστημα. (Εννοείται ότι δε γνωρίζουμε εξαρχής το τελικό πλήθος των ψηφίων του αριθμού στο νέο σύστημα, ώστε να δημιουργήσουμε το κατάλληλο πλήθος θέσεων).

Α) Μετατροπή από το δεκαδικό στο δυαδικό σύστημα. Έστω ότι θέλουμε να μετατρέψουμε τον δεκαδικό $N_{10} = 5310$ στο δυαδικό σύστημα.

ΠΗΛΙΚΟ		1	3	6	13	26	53
ΥΠΟΛΟΙΠΟ	**1**	**1**	**0**	**1**	**0**	**1**	X (διαίρεση δια 2)

Συνεπώς ο αριθμός $N_{10} = 5310$ στο δυαδικό σύστημα αρίθμησης είναι ο $N_2 = 1101012$.

Β) Μετατροπή από το δεκαδικό στο οκταδικό σύστημα. Να μετατραπεί ο αριθμός $N_{10} = 5310$ στο οκταδικό σύστημα.

ΠΗΛΙΚΟ		6	53
ΥΠΟΛΟΙΠΟ	**6**	**5**	X (διαίρεση δια 8)

Ο αριθμός $N_{10} = 5310$ στο οκταδικό σύστημα είναι ο $N_8 = 658$.

Γ) Μετατροπή από το δεκαδικό στο δεκαεξαδικό σύστημα. Να μετατραπεί ο αριθμός $N_{10} = 5310$ στο δεκαεξαδικό σύστημα.

ΠΗΛΙΚΟ		3	53
ΥΠΟΛΟΙΠΟ	**3**	**5**	X (διαίρεση δια 16)

Ο αριθμός στο δεκαεξαδικό σύστημα είναι ο $N_{16} = 3516$.

Επομένως ο αριθμός 53 του δεκαδικού συστήματος στα άλλα τρία συστήματα, που σχετίζονται με το δυαδικό σύστημα, έχει τους εξής ισοδύναμους αριθμούς: $N_2 = 110101_2$, $N_8 = 65_8$ και $N_{16} = 35_{16}$. Η επαλήθευση των αποτελεσμάτων γίνεται εύκολα μέσω της σχέσης 1-1:

$110101_2 = 1\cdot 2^5 + 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 32+16+0+4+0+1 = 53_{10}$

$65_8 = 6\cdot 8^1 + 5\cdot 8^0 = 48 + 5 = 53$

$35_{16} = 3\cdot 16_1 + 5\cdot 16_0 = 48 + 5 = 53$

Έστω ότι θέλουμε να μετατρέψουμε τον αριθμό **$N_{10}= 1973_{10}$** σε όλα τα γνωστά αριθμητικά συστήματα. Θα ξεκινήσουμε με τον υπολογισμό του ισοδύναμου δυαδικού αριθμού.

	1	3	7	15	30	61	123	246	493	986	1973
1	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	Χ (διαίρεση δια 2)

Ο αριθμός 1973 του δεκαδικού συστήματος στο δυαδικό σύστημα είναι ο **N_2** = **11110110101_2**. Για το οκταδικό σύστημα θα έχουμε:

	3	30	246	1973
3	**6**	**6**	**5**	Χ (διαίρεση δια 8)

Άρα ο αριθμός 1973 του δεκαδικού στο οκταδικό σύστημα είναι ο **$N_8= 3665_8$**.

Τέλος στο δεκαεξαδικό σύστημα θα έχουμε:

ΠΗΛΙΚΟ		7	123	1973
ΥΠΟΛΟΙΠΟ	**7**	**B**	**5**	Χ (διαίρεση δια 16)

Συνεπώς στο δεκαεξαδικό σύστημα αντιστοιχεί στον αριθμό **$N_{16}= 7B5_{16}$**.

Άρα ο αριθμός **1973_{10}** = **$7B5_{16}$** = **3665_8** = **11110110101_2**

1.4 Μετατροπή αριθμού από οποιοδήποτε σύστημα αρίθμησης στο δεκαδικό

Υπολογίζεται η τιμή του αριθμού στο δεκαδικό σύστημα αρίθμησης, χρησιμοποιώντας τις δυνάμεις της βάσης του συστήματος στην οποία βρίσκεται ο αριθμός. Η μετατροπή σ' αυτή την περίπτωση είναι πολύ απλή:

Α) Μετατροπή αριθμού από το δυαδικό στο δεκαδικό.

Να μετατραπεί ο δυαδικός **N_2= 101011** στον αντίστοιχο δεκαδικό **N_{10}**.

Ξεκινάμε πάντα από το λιγότερο σημαντικό ψηφίο -με τη μηδενική δύναμη- μέχρι να πάρουμε όλους τους όρους, αυξάνοντας τη δύναμη της βάσης συνεχώς κατά ένα, ή μετρούμε το πλήθος των ψηφίων και η δύναμη του μεγιστοβάθμιου όρου προκύπτει, αν αφαιρέσουμε ένα (1) από τον αριθμό που αντιστοιχεί στο πλήθος των ψηφίων, δηλαδή αν το πλήθος των ψηφίων είναι 6, τότε η δύναμη του μεγιστου βάθμιου όρου είναι το 5)

$N_2 = 101011 = 1\cdot 2^5+0\cdot 2^4+1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+1\cdot 2^0 = 32+0+8+0+2+1 = 43_{10}$

Β) Από οκταδικό σε δεκαδικό. Να μετατραπεί ο **$N_8 = 3765_8$** στον αντίστοιχο **N_{10}**

$N_8 = 3765 = 3\cdot 8^3+7\cdot 8^2+6\cdot 8^1+5\cdot 8^0 = 1536+448+48+5 = 2037_{10}$

Γ) Από δεκαεξαδικό σε δεκαδικό. Να μετατραπεί ο N_{16}= 1F στον αντίστοιχο N_{10}

$N_{16} = 1F = 1\cdot 16^1+F\cdot 16^0 = 1\cdot 16^1+15\cdot 16^0 = 16+15 = 31_{10}$

1.5 Μετατροπή από και προς οποιαδήποτε βάση αρίθμησης

Σημαντική παρατήρηση, πριν προχωρήσουμε σε οποιαδήποτε νέα μετατροπή, είναι να γνωρίζουμε ότι κάθε δεκαεξαδικός αριθμός ισοδυναμεί με ένα τετραψήφιο στο δυαδικό σύστημα και κάθε αριθμός του οκταδικού συστήματος ισοδυναμεί με ένα τριψήφιο στο δυαδικό σύστημα (*Πίνακας 1.1*)

ΔΕΚΑΔΙΚΟ	ΔΥΑΔΙΚΟ	ΟΚΤΑΔΙΚΟ	ΔΕΚΑΕΞΑΔΙΚΟ
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
—	—	—	—

Πίνακας 1.1. *Παράσταση αριθμών στα κύρια αριθμητικά συστήματα*

Ένας συνηθισμένος τρόπος, για να μετατρέψουμε έναν αριθμό που ανήκει σε οποιοδήποτε σύστημα αρίθμησης σ' έναν ισοδύναμο αριθμό διαφορετικού συστήματος, είναι να τον μετατρέψουμε αρχικά στον ισοδύναμο δεκαδικό και μετά στο επιθυμητό σύστημα.

Παράδειγμα:

Να μετατραπεί ο N_{16} = ABC στο οκταδικό σύστημα N_8 =;

N_{16}=== ABC === $A\cdot16^2+ B\cdot16^1+ C\cdot16^0=10\cdot16^2+11\cdot16^1+12\cdot16^0$===2560+176+12=2748

N_{16}=== $ABC \rightarrow N_{10}$===274810

	5	42	343	2748
5	**2**	**7**	**4**	X (διαίρεση δια 8)

Στο οκταδικό ο αριθμός είναι ο **N_8= 5274_8**

Ακολουθήσαμε τα βήματα, που έχουμε ήδη περιγράψει, δηλαδή τη μετατροπή στο δεκαδικό σύστημα και από εκεί στο οκταδικό: $N_{16} = ABC_{16} \rightarrow N_{10} = 2748_{10} \rightarrow N_8 = 5274_8$

1.6 Αναπαράσταση των θετικών και αρνητικών αριθμών στο δυαδικό σύστημα αρίθμησης

Θα εξετάσουμε τρεις διαφορετικούς τρόπους αναπαράστασης προσημασμένων αριθμών Την προσημασμένη παράσταση (Sign representation) Το συμπλήρωμα ως προς 1.

Το συμπλήρωμα ως προς 2.

Πώς μπορούμε να αποφασίσουμε για το ποια αναπαράσταση είναι καλύτερη;

Είναι προφανές ότι η καλύτερη αναπαράσταση θα πρέπει να καταλήγει σε απλούστερες και γρηγορότερες λειτουργίες. Μας ενδιαφέρουν δύο συγκεκριμένες λειτουργίες: Η μετατροπή του x σε –x και η πρόσθεση δύο προσημασμένων αριθμών, ή ο υπολογισμός της γενικής μορφής x + y.

1.6.1 Προσημασμένη παράσταση θετικών και αρνητικών δυαδικών αριθμών

Μέχρι στιγμής δεν εξετάσαμε τον τρόπο παράστασης των δυαδικών αριθμών, λαμβάνοντας υπόψη το πρόσημο τους, αλλά υποθέσαμε ότι όλες οι πράξεις αφορούσαν θετικούς αριθμούς. Διάφοροι τρόποι έχουν χρησιμοποιηθεί για την προσήμανση των δυαδικών αριθμών.

Συνήθως χρησιμοποιούμε ένα σύστημα προσημασμένων μεγεθών: Προσθέτουμε το + ή το – μπροστά από ένα μέγεθος για να προσδιορίσουμε το πρόσημό του.

Μπορούμε να το κάνουμε ίδιο και στο δυαδικό σύστημα, προσθέτοντας ένα επιπλέον δυαδικό ψηφίο πρόσημου μπροστά από τους αριθμούς. Συμβατικά λοιπόν το σύμβολο **0** αναπαριστά ένα θετικό αριθμό. και το σύμβολο **1** αναπαριστά έναν αρνητικό αριθμό.

Προσημασμένη παράσταση μεγέθους

Παρ.6ο:

$1101_2 = 13_{10}$ (4-bit μη προσημασμένος αριθμός)

01101 = $+13_{10}$ (5-bit θετικός αριθμός προσημασμένου μεγέθους)

11101 = -13_{10} (5-bit αρνητικός αριθμός προσημασμένου μεγέθους)

$0100_2 = 410$ (4-bit μη προσημασμένος αριθμός)

00100 = $+4_{10}$ (5-bit θετικός αριθμός προσημασμένου μεγέθους)

(5-bit αρνητικός αριθμός προσημασμένου
10100 = -410 μεγέθους)

Με αυτό τον τρόπο π.χ. το **-5** γράφεται σαν **10101** και το **+5** σαν **00101**. (Υποθέτουμε ότι χρησιμοποιούνται τέσσερα bit για την παράσταση της απόλυτης τιμής του αριθμού).

Παρόλο όμως ότι η παράσταση των προσημασμένων δυαδικών αριθμών με αυτό τον τρόπο ήταν πολύ εύκολη, δεν επικράτησε για δύο λόγους κυρίως α) γιατί έτσι γίνεται σπατάλη δυαδικών ψηφίων και β) για λειτουργικούς λόγους της Αριθμητικής Μονάδας (π.χ. η αφαίρεση δεν ήταν δυνατό να γίνει με τη βοήθεια της πρόσθεσης, αλλά απαιτούσε ξεχωριστές βαθμίδες). Οι περισσότερο χρησιμοποιούμενες μέθοδοι για την προ-σήμανση των δυαδικών αριθμών αναπτύσσονται παρακάτω.

Λειτουργίες προσημασμένων μεγεθών

Η μετατροπή ενός προσημασμένου αριθμού από θετικό σε αρνητικό είναι τετριμμένη: απλά αλλάζουμε το bit πρόσημου από 0 σε 1 και αντίστροφα.

Η πρόσθεση των αριθμών, όμως, είναι δύσκολη. Τα προσημασμένα μεγέθη είναι αυτά που κυρίως χρησιμοποιούμε και βασίζονται στη σύγκριση των προσήμων των δύο προσθετέων:

Αν έχουν το ίδιο πρόσημο, προσθέτουμε τα μεγέθη και κρατάμε το ίδιο πρόσημο. Αν έχουν διαφορετικά πρόσημα, αφαιρούμε το μικρότερο μέγεθος από το μεγαλύτερο. Το πρόσημο του μεγαλύτερου μεγέθους είναι και το πρόσημο του αποτελέσματος.

Αυτή η μέθοδος της αφαίρεσης, μάλλον θα μας οδηγήσει σε πολύπλοκο σχεδιασμό

		Αριθμός μετά την πρόσθεση και του δανεικού
		5 13 17
+3 7 9		
	διότι	**6 4 7**
+ - 6 4 7		
	ισχύει	**- 3 7 9**
- 2 6 8		**2 6 8**

Στη δεξιά αναπαράσταση και πάνω από τους αριθμούς σημειώνονται οι αριθμοί που προκύπτουν με-τά την πρόσθεση και της δανεικής μονάδας (δεκάδας). Έτσι σημειώνεται ο αριθμός 17, ο οποίος εμπεριέχει και μία δανεική δεκάδα από την προηγούμενη βαθμίδα, και αντίστοιχα το 13.

Προσοχή:Το17της δεξιάς στήλης, προκύπτει λόγω του δανεισμού από την προηγούμενη δεκάδα, ενώ αρχικά ήταν 7. Στη μεσαία στήλη το αρχικό 14(4+10 δανεικό από την προηγούμενη δεκάδα επίσης) γίνεται 14-1=13, καθώς δάνεισε δεξιότερα μια δεκάδα. Ομοίως το 5, που προκύπτει, πάνω από το 6 του περισσότερο σημαντικού ψηφίου, είναι απόρροια της πράξης 6-1 (πάλι με το ίδιο σκεπτικό με το προηγούμενο) λόγω του δανεισμού μιας δεκάδας, δεξιότερα. Άρα οι αριθμοί της πρώτης γραμμής, δεξιά, είναι οι αριθμοί που προκύπτουν, αν αφαιρέσουμε και το τυχόν δανεικό που μας χρειάζεται, για να υλοποιηθεί η αφαίρεση.

1.6.2. Παράσταση θετικών και αρνητικών δυαδικών αριθμών με συμπλήρωμα ως προς 1

Μια διαφορετική προσέγγιση είναι το συμπλήρωμα ως προς ένα κατά την οποία μετατρέπουμε έναν αριθμό από θετικό σε αρνητικό, συμπληρώνοντας κάθε δυαδικό ψηφίο του αριθμού (αλλάζοντας το 0 σε 1 ή το 1 σε 0 αντίστοιχα).

Σε ότι αφορά στο πρόσημο , κρατάμε τα δυαδικά ψηφία με την ίδια αναπαράσταση, όπως και προηγουμένως. Έτσι το 0 αντιστοιχεί σε θετικούς, και το 1 σε αρνητικούς αριθμούς. Το δυαδικό ψηφίο προσήμου συμπληρώνεται μαζί με τα υπόλοιπα δυαδικά ψηφία. Το ερώτημα που τίθεται είναι γιατί ονομάζεται «συμπλήρωμα ως προς ένα».

Το συμπλήρωμα ενός bit είναι ισοδύναμο, με την αφαίρεσή του από το 1. Έτσι 0 = 1 δηλαδή 1 - 0 = 1 και1 = 0 δηλαδή 1 - 1 = 0

Ομοίως, συμπληρώνοντας κάθε bit ενός n-bit αριθμού, είναι σαν να αφαιρούμε τον αριθμό αυτόν από τον αριθμό 2^n-1. Για παράδειγμα , έστω ότι θέλουμε να βρούμε τον αρνητικό του πενταψήφιου αριθμού 01101. Εδώ n=5, και 2^n-1 = 31_{10} = 11111_2.

Αφαιρώντας το 01101 από το 11111 έχουμε σαν αποτέλεσμα 10010:

	1	**1**	**1**	**1**	**1**
-	**0**	**1**	**1**	**0**	**1**
	1	**0**	**0**	**1**	**0**

Παράδειγμα 7ο:

Α)

1101_2 = 13_{10} (4-bit μη προσημασμένος αριθμός)

01101 = $+13_{10}$ (5-bit θετικός αριθμός με συμπλήρωμα ως προς ένα)

10010 = -13_{10} (5-bit αρνητικός αριθμός με συμπλήρωμα ως προς ένα)

0100_2 = 410 (4-bit μη προσημασμένος αριθμός)

00100 = $+4_{10}$ (5-bit θετικός αριθμός με συμπλήρωμα ως προς ένα)

11011 = -410 (5-bit αρνητικός αριθμός με συμπλήρωμα ως προς ένα)

1.6.3. Παράσταση θετικών και αρνητικών δυαδικών αριθμών με συμπλήρωμα ως προς 2

Συχνά μιλάμε για το «συμπλήρωμα ως προς δύο» ενός αριθμού. Αυτή είναι μια «μπερδεμένη φράση», αλλά συνήθως συσχετίζεται με την αλλαγή του πρόσημου κάποιου αριθμού.

Όπως και στην παράσταση με συμπλήρωμα ως προς 1 έτσι και εδώ, εάν το πρώτο δυαδικό ψηφίο του αριθμού είναι «1» ο αριθμός είναι αρνητικός, ενώ εάν είναι «0» ο αριθμός είναι θετικός, με τη διαφορά ότι στην αναπαράσταση αυτή και το πρόσημο παίζει ρόλο στην αξία του αριθμού.

Για να μετατρέψουμε έναν αριθμό σε αρνητικό, συμπληρώνουμε κάθε δυαδικό ψηφίο (όπως στο συμπλήρωμα ως προς ένα) κι έπειτα προσθέτουμε 1 στο λιγότερο σημαντικό ψηφίο.

Στον *πίνακα1.5* παραθέτουμε ορισμένα παραδείγματα δυαδικών αριθμών (πρώτη στήλη) μαζί τον αντίστοιχο δεκαδικό αριθμό που παριστάνουν (δεύτερη στήλη).

Δύο άλλοι, ισοδύναμοι τρόποι για να μετατρέψουμε αριθμούς, στο συμπλήρωμα ως προς δύο είναι:

α) Να αφαιρέσουμε έναν αριθμό των n-bit από τον αριθμό 2^n, π.χ.

	1	**0**	**0**	**0**	**0**	**0**			**1**	**0**	**0**	**0**	**0**	**0**
-		**0**	**1**	**1**	**0**	**1**	**(+13$_{10}$)**	-		**0**	**0**	**1**	**0**	**0**	**(+4$_{10}$)**
	1	**0**	**0**	**1**	**1**	**(-13$_{10}$)**		**1**	**1**	**1**	**0**	**0**	**(-4$_{10}$)**

β) Να συμπληρώσουμε όλα τα bits που βρίσκονται αριστερά από τον δεξιότερο 1.

01101 = +13_{10} (ένας θετικός αριθμός σε συμπλήρωμα ως προς δύο)

10011 = - 13_{10} (ένας αρνητικός αριθμός σε συμπλήρωμα ως προς δύο)

00100 = + 4_{10} (ένας θετικός αριθμός σε συμπλήρωμα ως προς δύο)

11100 = - 4_{10} (ένας αρνητικός αριθμός σε συμπλήρωμα ως προς δύο)

Παράδειγμα 8ο:

Α)

1101_2 = 13_{10} (4-bit μη προσημασμένος αριθμός)

0110 1 = +13_{10} (5-bit θετικός αριθμός με συμπλήρωμα ως προς δύο)

1001 0 = - 13_{10} (5-bit αρνητικός αριθμός με *συμπλήρωμα ως προς ένα*)

1001 1 = - 13_{10} (5-bit αρνητικός αριθμός με συμπλήρωμα ως προς δύο)

010 0 = 410 (4-bit μη προσημασμένος αριθμός)

0010 0 = + 4_{10} (5-bit θετικός αριθμός με συμπλήρωμα ως προς δύο)

11011= - 4_{10} (5-bit αρνητικός αριθμός με *συμπλήρωμα ως προς ένα*)

11100= - 4_{10} (5-bit αρνητικός αριθμός με συμπλήρωμα ως προς δύο)

Β)

Προσημασμένες δυαδικές παραστάσεις με συμπλήρωμα ως προς 2	**Αντίστοιχοι δεκαδικοί αριθμοί**
01001100	+76
01111111	+127
10000000	-128
11111111	-1
00000000	+0
10101010	-86

Πίνακας 1.5. *Προσημασμένοι δυαδικοί αριθμοί με συμπλήρωμα ως προς2*

Εάν υποθέσουμε και εδώ ότι έχουμε δυαδικές παραστάσεις με 6 δυαδικά ψηφία, ο μεγαλύτερος θετικός αριθμός που δύναται να γραφεί είναι ο 011111 (+31) ενώ ο μικρότερος αρνητικός είναι ο 100000 (-32). Δηλαδή με 6 δυαδικά ψηφία μπορούμε να παραστήσουμε τους αριθμούς από -32 έως +31.

Για «n» δυαδικά ψηφία ο μεγαλύτερος θετικός θα είναι ο 2^{n-1}-1 και ο μικρότερος αρνητικός θα είναι ο-2^{n-1}.

ΚΕΦΑΛΑΙΟ 2ο :

Εισαγωγή στους αριθμούς κινητής υποδιαστολής

2.1 Τι είναι οι αριθμοί κινητής υποδιαστολής

Υπάρχουν διάφοροι τρόποι να αναπαρασταθούν οι πραγματικοί αριθμοί στους υπολογιστές. Ένας τρόπος είναι οι αριθμοί κινητής υποδιαστολής. Η κινητή υποδιαστολή δεν είναι προκαθορισμένη, αλλά μεταβλητή και αυτόματα προσαρμόσιμη στις μεταβαλλόμενες απαιτήσεις των υπολογισμών. Σε μια τέτοια περίπτωση, η θέση της δυαδικής υποδιαστολής λέμε ότι μετακινείται (floats) και οι αριθμοί αυτή ονομάζονται *αριθμοί κινητής υποδιαστολής.* Αυτό τους ξεχωρίζει από τους αριθμούς σταθερής υποδιαστολής, στους οποίους η υποδιαστολή βρίσκεται πάντα στην ίδια θέση.

Για την αναπαράσταση των αριθμών κινητής υποδιαστολής χρησιμοποιείται η επιστημονική σημειογραφία. Στην οποία οι αριθμοί γράφονται στη μορφή d.dddd x 10^{exp} όπου d είναι τα δεκαδικά ψηφία η βάση 10 και exp ο εκθέτης. Μετατοπίζουμε στην ουσία την υποδιαστολή σε μια θέση βολική και χρησιμοποιούμε δυνάμεις του 10 για να ξέρουμε τη θέση αυτής της δεκαδικής υποδιαστολής.

Για παράδειγμα :

- Ο αριθμός 123.456 θα μπορούσε να αναπαρασταθεί ως :

  1.23456×10^{2}.

- Στον δεκαεξαδικό, το νούμερο 123.abc επίσης θα μπορούσε να αναπαρασταθεί ως : $1.23abc \times 16^{2}$.

2.2 Γιατί χρησιμοποιούμε αριθμούς κινητής υποδιαστολής

Θεωρήστε το εύρος των τιμών ο οποίες είναι αναπαραστάσιμες σε μια προσημασμένη μορφή σταθερής υποδιαστολής των 32 bit. Εάν ερμηνευτούν αυτές οι τιμές ως ακέραιοι αριθμοί, το εύρος των τιμών αυτού του διαστήματος είναι από 0 έως περίπου ±2.15 x 10^{9}. Εάν τους θεωρήσουμε ως κλάσματα, το εύρος είναι περίπου από ±4.55 x10^{-10} έως ±1. Κανένα από αυτά τα δύο εύρη δεν είναι ικανοποιητικό για επιστημονικούς υπολογισμούς, οι οποίοι μπορούν να εμπεριέχουν παραμέτρους όπως ο αριθμός Avogardo (6.0247 x 10^{23} $mole^{-1}$), ή η σταθερά του Planck (6.6254 x 10^{-27} erg·s). Επομένως, χρειαζόμαστε έναν βολικό τρόπο αναπαράστασης αμφότερων των πολύ μεγάλων ακεραίων και των πολύ μικρών κλασμάτων. Κάτι το οποίο πετυχαίνουμε χρησιμοποιώντας αριθμούς κινητής υποδιαστολής.

Με τους αριθμούς κινητής υποδιαστολής μπορούμε να αναπαραστήσουμε εύκολα τα νούμερα όπως 1.000.000.000.000 ή 0.0000000000001.

Οι αριθμοί κινητής υποδιαστολής και η αναπαράσταση χρησιμοποιούνται συνήθως για υπολογισμούς που απαιτούν μεγάλες ακρίβειες. Απαλλάσσουν των προγραμματιστή από τον πονοκέφαλο της κλίμακας και μπορούν να αποθηκευτούν μεγάλη αριθμοί σε σχετικά λίγα bits.

Το πλεονέκτημα είναι ότι μπορεί να χωρέσουν σε σχετικά λίγα bits μεγάλοι αριθμοί αλλά το μειονέκτημα αυτής της αναπαράστασης είναι ότι δεν μπορούμε να έχουμε ν-1 σημαντικά bits όπως στο σύστημα αναπαράστασης των ακεραίων.

2.3 Η μορφή των αριθμών κινητής υποδιαστολής

Ξεκινούμε με γενικές υποθέσεις, όσον αφορά τη μορφή και το μέγεθος των αριθμών κινητής υποδιαστολής στο δεκαδικό σύστημα και κατόπιν συσχετίζουμε τη μορφή αυτή με μια αντίστοιχη δυαδική αναπαράσταση. Μια χρήσιμη μορφή είναι η

$$\pm X_1.X_2X_3X_4X_5X_6X_7 \times 10^{\pm Y1Y2} \quad (1.1)$$

όπου τα $X_ι$ και $Y_ι$ είναι δεκαδικά ψηφία. Ο αριθμός των σημαντικών ψηφίων(7) και το εύρος εκθέτη (±99) ανταποκρίνονται σε μια ευρεία κλίμακα επιστημονικών υπολογισμών. Η προσέγγιση αυτή της ακρίβειας του δεκαδικού μέρους (mantissa) και του εύρους του παράγοντα κλίμακας , είναι δυνατή σε μια δυαδική αναπαράσταση, η οποία καταλαμβάνει 32 bits, κάτι το οποίο είναι ένα τυποποιημένο μέγεθος λέξης ενός υπολογιστή.

2.4 Πως αποθηκεύονται οι αριθμοί κινητής υποδιαστολής

Οι αριθμοί κινητής υποδιαστολής αποθηκεύονται σε 3 πεδία : Το πρόσημο (Sign), τον εκθέτη (Exponent) και το σημαντικό μέρος (Mantissa).

S	Exp	Fraction (or Mantissa)

Η υπονοούμενη βάση είναι το 2. Οπότε δεν περιλαμβάνεται στη αναπαράσταση αλλά υπονοείται. Από αυτή την μορφή μπορούμε να υπολογίσουμε την πραγματική τιμή του αριθμού κινητής υποδιαστολής. Όπως θα δούμε πιο κάτω. Σε αυτή την αναπαράσταση για να βρούμε την δεκαδική του τιμή αρκεί να υπολογίσουμε την σχέση:

$$(-1)S * \text{mantissax } 2^{\text{exponent}}$$

2.4.1 Πρόσημο (SIGN)

Στους αριθμούς κινητής υποδιαστολής το πρώτο bit δηλώνει το πρόσημου του αριθμού που είναι αποθηκευμένο στα επόμενα bit. Εάν λοιπόν το bit αυτό είναι 1 ο αριθμός είναι αρνητικός αλλιώς εάν είναι 0 τότε ο αριθμός είναι θετικός.

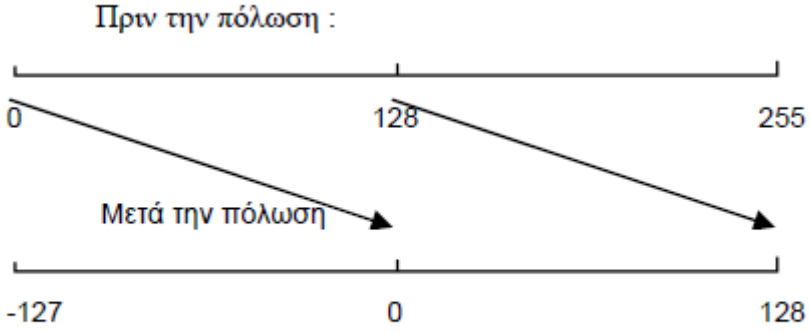
2.4.2 Εκθέτης (EXPONENT)

Ο εκθέτης αποθηκεύεται μετά το πρόσημο. Στην αναπαράσταση του εκθέτη χρησιμοποιείται η *αναπαράσταση με πόλωση*. Η πόλωση αυτή ισούται με

$$(2^{k-1}-1)$$

Όπου k είναι ο αριθμός των bit στον δυαδικό εκθέτη. Η πόλωση αυτή προστίθεται στο εκθέτη όταν είναι να αποθηκευτή σε αυτή την μορφή. Στην αντίστροφη μετατροπή αυτή η τιμή αφαιρείται από τον εκθέτη. Και έτσι παίρνουμε την πραγματική τιμή του εκθέτη. Για παράδειγμα στην περίπτωση που ο εκθέτης είναι 8 bits η τιμή αυτή ισούται με $(2^{8-1}-1)$ και είναι το 127. Έτσι λοιπόν αν θεωρήσουμε για παράδειγμα ότι έχουμε τον αριθμό X με εκθέτη το -20. Ο αριθμός -20 είναι στον δυαδικό -10100. Μετά την πόλωση τους -20 + 127 = 107. Τελικά ο εκθέτης θα πάρει την τιμή 1101011.

Στην περίπτωση των 8 bit η τιμή πόλωσης είναι προφανώς το 127, όπου τους δίνει τους αριθμούς από 0 έως 255. Και με την πόλωση η τιμή του εκθέτη βρίσκεται στο διάστημα -127 128.



Σχήμα 2. Πολωμένη αναπαράσταση για εκθέτη των 8 bit

Με αυτό τον τρόπο έχουμε την δυνατότητα να αναπαριστάνουμε και εκθέτες με αρνητικό πρόσημο. Τους ένα από τα **πλεονεκτήματα** τους πολωμένης αναπαράστασης είναι ότι οι μη αρνητική αριθμοί κινητής υποδιαστολής μπορούν να θεωρούνται ακέραιοι για σκοπούς σύγκρισης.

2.4.3 Σημαντικό μέρος (MANTISSA)

Το πεδίο αμέσως μετά τον εκθέτη ονομάζεται mantissa ή αλλιώς σημαντικό μέρος. Στην τυπική μορφή των 32 bit αποτελείται από 23 bit. Στην ουσία όμως περιέχει και ένα κρυφό (hidden) bit που υπονοείται και δεν χρειάζεται να αποθηκευτή κερδίζοντας έτσι από τον χώρο. Κατά την αποθήκευση τους το σημαντικό μέρος με μετατοπίσεις και αύξηση του εκθέτη γίνεται κανονικοποίηση (*βλ. ΚΕΦ 1.5*). Δηλαδή στην ουσία φέρνουμε τον αριθμό στην μορφή:

$$(-1)S \times 1.bbbb \times 2^{exponent}$$

Όπως παρατηρείτε το πρώτο bit είναι 1 (ένα) και τα υπόλοιπα bit είναι μετά την υποδιαστολή Μας βολεύει όμως για να γλιτώσουμε και από ένα bit επιπλέον και επίσης οι πράξεις με τους αριθμούς απλοποιούνται με αυτό τον τρόπο.
Οποιοσδήποτε αριθμός κινητής υποδιαστολής μπορεί να εκφραστεί με πολλούς τρόπους . Για παράδειγμα οι εξής αριθμοί είναι ισοδύναμοι :

$$0.011 \times 2^{4}$$

$$1.100 \times 2^{-2}$$

$$0.110 \times 2^{-1}$$

Αυτό επιτυγχάνεται με την μετατόπιση και αύξηση η μείωση του εκθέτη όπως θα δείτε.

2.5 Κανονικοποίηση

Ένας **Κανονικοποιημένος Αριθμός** είναι ένας αριθμός στον οποίο το πιο σημαντικό ψηφίο του σημαντικού μέρους είναι **μη μηδενικό**. Για την αναπαράσταση με βάση το 2, ένας κανονικοποιημένος αριθμός είναι εκείνος, στον οποίο το πιο σημαντικό bit του σημαντικού μέρους είναι 1. Η τυπική σύμβαση είναι ότι υπάρχει ένα bit στα αριστερά του σημείου της υποδιαστολής. Έτσι ένας κανονικοποιημένος μη μηδενικός αριθμός είναι εκείνος με τη μορφή (*βλ. 1.4*)

Όπου **b** είναι οποιοδήποτε δυαδικό ψηφίο (0 ή 1). Επειδή το πιο σημαντικό bit είναι πάντοτε 1, δεν υπάρχει ανάγκη να το αποθηκεύσουμε, αλλά υπονοείται. Έτσι το πεδίο των 23 bit χρησιμοποιείται για να αποθηκευτεί ένα σημαντικό μέρος με 24 bit, με τιμή στο ημιανοικτό διάστημα [1,2).

Εάν δοθεί ένας αριθμός που δεν είναι κανονικοποιημένος μπορεί να κανονικοποιηθεί μετατοπίζοντας την υποδιαστολή στα δεξιά του πιο αριστερού bit με τιμή 1 και ρυθμίζοντας αντίστοιχα τον εκθέτη.

Για παράδειγμα ας κανονικοποιήσουμε τους παρακάτω 2 αριθμούς:

α) 0.00011×2^{4}

$0.00011 \times 2^{4} = 0.0011 \times 2^{3} = 0.011 \times 2^{2} = 0.11 \times 2^{1} = 1.1 \times 2^{0}$

β) 10010.001×2^{0}

$10010.001 \times 2^{0} = 1001.0001 \times 2^{1} = 100.10001 \times 2^{2} = 10.010001 \times 2^{3} = 1.0010001 \times 2^{4}$

Όπως βλέπετε αυξάνεται ή μειώνεται ο εκθέτης και ταυτόχρονα μετατοπίζεται αντίστοιχα δεξιά και αριστερά ο εκθέτης. Στην ουσία ο εκθέτης δηλώνει σε πιο σημείο βρίσκεται η υποδιαστολή.

2.6 Συμβιβασμός μεταξύ περιοχής τιμής και ακρίβειας

Είναι σημαντικό να παρατηρήσουμε, ότι δεν αναπαριστούμε περισσότερες μεμονωμένες τιμές με την σημειογραφία κινητής υποδιαστολής. Ο μέγιστος αριθμός διαφορετικών τιμών που μπορεί να αναπαρασταθεί με 32 bit είναι πάντοτε 2^{32}. Εκείνο που πετύχαμε είναι να "απλώσουμε" τους αριθμούς αυτούς σε δυο διαστήματα ένα θετικό και ένα αρνητικό. Υπάρχει ένας συμβιβασμός μεταξύ περιοχής τιμών και ακρίβειας. Υπάρχουν 8 bit αφιερωμένα στην αναπαράσταση του εκθέτη και 23 bit για την αναπαράσταση του σημαντικού μέρους. Αν αυξήσουμε τον αριθμό των bit στον εκθέτη, επεκτείνουμε την περιοχή τιμών των αριθμών που μπορεί να εκφραστούν. Επειδή όμως μόνο ένας σταθερός αριθμός διαφορετικών τιμών μπορεί να εκφραστεί, έχουμε μειώσει την "πυκνότητα" εκείνων των αριθμών και κατά συνέπεια την ακρίβεια. Ο μόνος τρόπος για να αυξηθεί τόσο η περιοχή τιμών όσο και η ακρίβεια είναι να χρησιμοποιηθούν περισσότερα bit. Έτσι, οι περισσότεροι υπολογιστές προσφέρουν τουλάχιστον αριθμούς απλής ακρίβειας και αριθμούς διπλής ακρίβειας. Για παράδειγμα μια μορφή απλής ακρίβειας θα μπορούσε να είναι με 32 bit και η μορφή διπλής ακρίβειας με 64 bit.

2.7 Συμβιβασμός μεταξύ του αριθμού, του εκθέτη και σημαντικού μέρους

Υπάρχει ένας συμβιβασμός μεταξύ του αριθμού των bit στον εκθέτη και των bit στο σημαντικό μέρος. Τα πράγματα όμως είναι ακόμη πιο σύνθετα. Η υπονοούμενη βάση του εκθέτη δεν υπάρχει ανάγκη να είναι το 2. Η αρχιτεκτονική IBM S/390, για παράδειγμα, χρησιμοποιεί βάση το 16. Η μορφή αυτή αποτελείται από έναν εκθέτη με 7 bit και σημαντικό μέρος με 24 bit.

Το πλεονέκτημα χρήσης ενός μεγαλύτερου εκθέτη είναι ότι μπορεί να επιτευχθεί αναπαράσταση μεγαλύτερης περιοχής τιμών για τον ίδιο αριθμό bit του εκθέτη. Όμως, δεν έχουμε αυξήσει τον αριθμό των διαφορετικών τιμών οι οποίες μπορούν να αναπαρασταθούν. Έτσι, για μια σταθερή μορφή, μια μεγαλύτερη βάση του εκθέτη δίνει μεγαλύτερη περιοχή τιμών, με αντάλλαγμα την μικρότερη ακρίβεια.

ΚΕΦΑΛΑΙΟ 3ο :

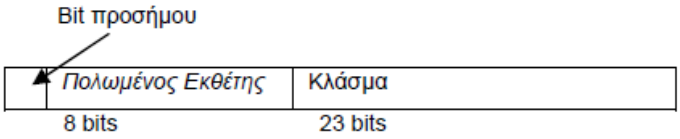
Το πρότυπο IEEE για την αναπαράσταση των αριθμών κινητής υποδιαστολής

3.1 Το πρότυπο 754 του IEEE για την αναπαράσταση

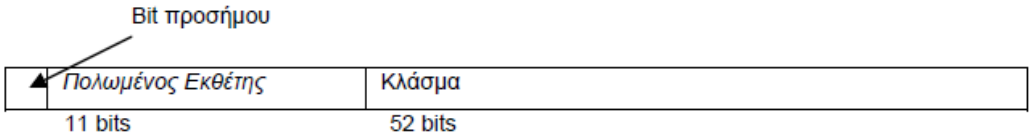
Η IEEE έχει σαν στόχο την κατασκευή διεθνών προτύπων. Το συγκεκριμένο πρότυπο 754 αναπτύχθηκε για να διευκολυνθεί η φορητότητα των προγραμμάτων από τον έναν επεξεργαστεί στον άλλον, και για να προωθηθεί η ανάπτυξη σύνθετων προγραμμάτων με προσανατολισμό τους υπολογισμούς. Το πρότυπο έχει γίνει ευρέως αποδεκτό και χρησιμοποιείται ουσιαστικά σε όλους τους σύγχρονους επεξεργαστές και τους αριθμητικούς συνεπεξεργαστές.

3.2 Μορφές αριθμών κινητής υποδιαστολής

Το πρότυπο IEEE 754 ορίζει 2 βασικές μορφές και 2 εκτεταμένες μορφές . Σύμφωνα με το πρότυπο οι 2 βασικές μορφές είναι η απλής ακρίβειας (single precision) και διπλής ακρίβειας (double precision). Οι μορφές είναι οι απλή-εκτεταμένη (single-extended precison) και διπλή-εκτεταμένη (double-extended precision).



Σχήμα 3. SINGLE PRECISION
Μορφή απλής ακρίβειας



Σχήμα 4. DOUBLE PRECISION
Μορφή διπλής ακρίβειας

Η απλή ακρίβεια αποτελείται από 1 bit πρόσημο, 8 bit εκθέτη και 23 bit σημαντικό μέρος. Η διπλή ακρίβεια έχει 11 bit στον εκθέτη, και 52 bit στο σημαντικό μέρος.

Η μορφή των εκτεταμένων μορφών εξαρτάται από την συγκεκριμένη υλοποίηση. Οι εκτεταμένες μορφές περιλαμβάνουν επιπρόσθετα bit στον εκθέτη (εκτεταμένη περιοχή τιμών) και στο σημαντικό μέρος (εκτεταμένη ακρίβεια προορίζονται **δε** για χρήση σε ενδιάμεσους υπολογισμούς.

Με τη μεγαλύτερη ακρίβεια τους, οι εκτεταμένες μορφές μειώνουν την πιθανότητα εμφάνισης ενός τελικού αποτελέσματος το οποίο έχει υποβαθμιστεί από υπερβολικό σφάλμα στρογγυλοποίησης.

Με τη μεγαλύτερη περιοχή τιμών τους, μειώνουν επίσης την πιθανότητα να ακυρωθεί ένας ενδιάμεσος υπολογισμός λόγω υπερχείλισης, ενώ το τελικό αποτέλεσμα θα μπορούσε να αναπαρασταθεί με μια βασική μορφή.

Ένα ακόμη κίνητρο για την χρήση αυτής της μορφής απλής επέκτασης είναι ότι έχει κάποια από τα πλεονεκτήματα μιας μορφής διπλής ακρίβειας, χωρίς όμως το κόστος σε χρόνο που συνήθως απαιτεί η υψηλότερη ακρίβεια.

Ο παρακάτω πίνακας δίνει περιληπτικά τα χαρακτηριστικά των τεσσάρων μορφών :

ΠΑΡΑΜΕΤΡΟΣ	**ΑΠΛΗ ΜΟΡΦΗ**	**ΑΠΛΗ ΕΚΤΕΤΑΜΕΝΗ ΜΟΡΦΗ**	**ΔΙΠΛΗ ΜΟΡΦΗ**	**ΔΙΠΛΗ ΕΚΤΕΤΑΜΕΝΗ ΜΟΡΦΗ**
Εύρος λέξης (bit)	32	≥ 43	64	≥ 79
Εύρος εκθέτη (bit)	8	≥ 11	11	≥ 15
Πόλωση εκθέτη	127	*ακαθόριστη*	1023	*ακαθόριστη*
Μέγιστος εκθέτης	127	≥ 1023	1023	≥ 16383
Ελάχιστος εκθέτης	-126	≤-1022	-1022	≤-16382
Περιοχή τιμών αριθμών (βάση 10)	$10^{-38} – 10^{+38}$	*ακαθόριστη*	$10^{-308} – 10^{+308}$	*ακαθόριστη*
Εύρος του σημαντικού μέρους (bit)	23	≥ 31	52	≥ 63
Πλήθος εκθετών	254	*ακαθόριστη*	2046	*ακαθόριστη*
Πλήθος κλασμάτων	2^{23}	*ακαθόριστη*	2^{52}	*ακαθόριστη*
Πλήθος τιμών	1.98×2^{31}	*ακαθόριστη*	1.99×2^{63}	*ακαθόριστη*

3.3 Μετατροπή αριθμών κινητής υποδιαστολής

Η μετατροπή των αριθμών κινητής υποδιαστολής από δεκαδικό σε δυαδικό και αντίστροφα είναι ένα αρκετά σημαντικό θέμα. Θα μελετήσουμε την μετατροπής από δεκαδικό σε δυαδικό και από δυαδικό σε δεκαδικό των αριθμών κινητής υποδιαστολής.

3.3.1 Μετατροπή από δεκαδικό σε δυαδικό

Η μετατροπή από 10δικό σε 2δικό σύστημα γίνεται με τον εξής τρόπο :

1) Γράφουμε ο αριθμός στην μορφή:

$$\text{mantissa}_{10} \times 2^{0}$$

όπου η mantissa είναι στο δεκαδικό σύστημα.

2) Μετατρέπουμε την mantissa από δεκαδικό στον δυαδικό με τον κλασικό τρόπο μετατροπής που ξέρουμε δηλαδή με διαδοχικές διαιρέσεις το ακέραιο μέρος και με διαδοχικούς πολλαπλασιασμούς το δεκαδικό μέρος και μας προκύπτει ένας αριθμός της μορφής:

$$bbbbb.bbbbb \times 2^{0}$$

3) Στη συνέχεια κανονικοποιούμε ο αριθμός όπως είχαμε αναφέρει πριν (*βλ. 2.5*) και το φέρνουμε στην μορφή (1.4).
4) Προσθέτουμε την τιμή πόλωσης στον εκθέτη.
5) Μετατρέπουμε τον εκθέτη από δεκαδικό σε δυαδικό με τον κλασικό τρόπο διαδοχικών διαιρέσεων.

6) Καθορίζουμε το πρόσημο του αριθμού και εάν το νούμερο είναι θετικό το bit παίρνει τιμή 0 και εάν είναι αρνητικό 1 (*βλ. 1.6.1*).
7) Στο τέλος έχουμε της τιμές :

S πρόσημο
E εκθέτης
M σημαντικό μέρος

Της οποίες γράφουμε με την μορφή IEEE.(*βλ. Σχήματα 3 και 4*)

3.3.2 Μερικά παραδείγματα μετατροπής από δεκαδικό σε δυαδικό

Ας δούμε μερικά παραδείγματα μετατροπής από δεκαδικό σε single μορφή για να γίνει πιο κατανοητό ο τρόπος μετατροπής.

ΠΑΡΑΔΕΙΓΜΑ 1:

Έστω ότι έχουμε τον αριθμό 30.

1) Γράφουμε ως εξής:

$$30 \times 2^{0}$$

2) Μετατρέπουμε την mantissa και έχουμε :

$$11110.0 \times 2^{0}$$

(παρατηρήστε ότι δεν έχουμε δεκαδικό μέρος οπότε αρχικά έχουμε μηδέν).

3) Κανονικοποιούμε τον αριθμό (*βλ ΚΕΦ 1.5*) και έχουμε:

$$1.111 \times 2^{4}$$

4) Ο εκθέτης μας είναι 4. προσθέτουμε την τιμή πόλωσης που είναι το 127 και έχουμε 131.
5) Μετατρέπουμε τον δεκαδικό 131 σε δυαδικό και έχουμε :

$$1.111 \times 2^{10000011}$$

6) Καθορίζουμε το πρόσημο του αριθμού. Εφόσον είναι ο αριθμός με θετικό άρα το πρόσημο θα είναι 0.
7) Στο τέλος γράφουμε το νούμερο μας στην μορφή IEEE ως εξής :

0	10000011	11100000000000000000000

ΠΑΡΑΔΕΙΓΜΑ 3:

Έστω ότι έχουμε τον αριθμό $1,25 \times 10^{100}$

Εδώ λοιπόν το πρόβλημα μας είναι ότι πάλι έχουμε έναν δεκαδικό εκθέτη αλλά αυτή την φορά ο εκθέτης είναι αρκετά μεγάλος. Θα μπορούσαμε να έχουμε έναν εκθέτη 10^{4000}. Αλλά για λόγους απλότητας επιλέξαμε αυτόν τον εκθέτη.

Εδώ λοιπόν μια δυνατή τεχνική που μπορεί να χρησιμοποιηθεί είναι να υπολογιστεί ξεχωριστά η βάση και χωριστά ο δεκαδικός εκθέτης και δυαδικός εκθέτης.

Η διαδικασία του υπολογισμού είναι :

```
        ΟΣΟ (δεκαδικός εκθέτης >0)
                ΕΑΝ δεκαδικός εκθέτης ΘΕΤΙΚΟΣ
        ΒΑΣΗ=ΒΑΣΗ/2
        δυαδικός εκθέτης + 1
                ΑΛΛΙΩΣ
ΒΑΣΗ=ΒΑΣΗ*2

        δυαδικός εκθέτης - 1
                ΕΑΝ δεκαδικός εκθέτης ΘΕΤΙΚΟΣ και ΒΑΣΗ < 1
                δεκαδικός εκθέτης-1
                ΕΑΝ δεκαδικός εκθέτης ΑΡΝΗΤΙΚΟΣ και ΒΑΣΗ >10
                δεκαδικός εκθέτης+1
ΕΠΑΝΑΛΗΨΗ
```

Στο τέλος παίρνουμε τον αριθμό : $5,714936956 * 2^{330}$. Παρατηρήστε ότι $5,714936956 * 2^{330}$ μας κάνει $1,25 \times 10^{100}$. Δηλαδή στο τέλος όλης της διαδικασία παίρνουμε έναν αριθμό με βάση το 2. Έχουμε δηλαδή έτοιμο τον εκθέτη. Αρκεί να κάνουμε της κλασικές διαδικασίες για να βρούμε τον αριθμό.

Για να γίνει όμως πιο κατανοητή όλη η διαδικασία ας το κάνουμε βήμα βήμα: Πρώτα κάνουμε της πράξεις για να βρούμε τον τελικό αριθμό 5,714936956 * 2^{330}. Έχουμε λοιπόν των παρακάτω πίνακα με τα βήματα εκτέλεσης της διαδικασίας :

ΒΗΜΑ	**Βάση**	**Δεκαδικός εκθέτης**	**Δυαδικός Εκθέτης**
0	1,25	100	0
1	6,25	99	1
2	3,125	99	2
3	1,5625	99	3
4	7,8125	98	4
5	3,90625	98	5
6	1,953125	98	6
7	9,765625	97	7
8	4,8828125	97	8
9	2,44140625	97	9
10	1,220703125	97	10
……	……	……	……
……	……	……	……
……	……	……	……
320	5,852095443	3	320
321	2,926047722	3	321
322	1,463023861	3	322
323	7,315119304	2	323
324	3,657559652	2	324
325	1,828779826	2	325
326	9,14389913	1	326
327	4,571949565	1	327
328	2,285974783	1	328
329	1,142987391	1	329
330	5,714936956	0	330

Στο τέλος παίρνουμε δεκαδικό εκθέτη 0, δυαδικό εκθέτη 330 και βάση 5,714936956. Στην ουσία σε κάθε βήμα διαιρούμε η πολλαπλασιάζουμε την βάση, εάν δηλαδή ο εκθέτης είναι αρνητικός πολλαπλασιάζουμε την βάση με 2 εάν είναι θετικός την διαιρούμε. Σε κάθε βήμα αυξάνουμε ή μειώνουμε αντίστοιχα τον δυαδικό εκθέτη πάντοτε. Για τον δεκαδικό εκθέτη εάν είναι θετικός και η βάση γίνει μικρότερη του 1, τότε ο δεκαδικός εκθέτης μειώνεται και η βάση πολλαπλασιάζετε με το 10. Εάν ο δεκαδικός εκθέτης είναι αρνητικός και η βάση είναι μεγαλύτερη του 10 τότε ο δεκαδικός εκθέτης μειώνεται και η βάση διαιρείται με 2. Έχουμε λοιπόν στο τέλος όλης αυτής της διαδικασίας την βάση 5,714936956 και τον εκθέτη 330. Μετατρέπουμε την βάση στο δεκαδικό ξεχωριστά το ακέραιο μέρος και ξεχωριστά το δεκαδικό μέρος και έχουμε :

$$5_{10} = 101$$

$0{,}714936956_{10}$ = 0.10110111000001100001101110111111….

Έχουμε τελικά 101. 10110111000001100001101… με δυαδικό εκθέτη 330. Τον κανονικοποιούμε και έχουμε :
1.01101101110000011000011101… και εκθέτη 332. Ο δυαδικό αριθμός αυτός ισούται με 1,428734239 περίπου. Αυτό είναι ένα πρόβλημα ακρίβειας και μερικούς αριθμούς μετά την μετατροπή χρειάζονται στρογγυλοποίηση. Εν τέλη έχουμε:

0	10101001011	01101101110000011000011011101111100111110100010111 00

Προσοχή όμως διότι ο αριθμός είναι σε διπλή ακρίβεια και όχι σε απλή. Οπότε έχει διαφορετική πόλωση και όπως βλέπουμε περισσότερα bits για αποθήκευση.

ΠΑΡΑΔΕΙΓΜΑ 4:

Ας δούμε και έναν αριθμό με αρνητικό εκθέτη και αρνητικό πρόσημο. Ο αριθμός : $-1{,}25 \times 10^{-5}$
Πρώτα αντιστρέφουμε το πρόσημο του αριθμού. Με την παραπάνω διαδικασία έχουμε τα παρακάτω:

ΒΗΜΑ	**Βάση**	**Δεκαδικός εκθέτης**	**Δυαδικός Εκθέτης**
0	1,25	-5	0
1	2,5	-5	-1
2	5	-5	-2
3	1	-4	-3
4	2	-4	-4
5	4	-4	-5
6	8	-4	-6
7	1,6	-3	-7
8	3,2	-3	-8
9	6,4	-3	-9
10	1,28	-2	-10
11	2,56	-2	-11
12	5,12	-2	-12
13	1,024	-1	-13
14	2,048	-1	-14
15	4,096	-1	-15
16	8,192	-1	-16
17	1,6384	0	-17

Έχουμε ττους αριθμούς 1,6384 και δυαδικό εκθέτη -17. Με την μετατροπή αυτών έχουμε :

$$1{,}6384_{10} = 1.10100011011011100010111$$

Ο εκθέτης μετά την πόλωση -17+127=110 ο οποίος στο δυαδικό μας κάνει 1101110.

Άρα τελικά έχουμε :

1	1101110	10100011011011100010111

3.3.3 Μετατροπή από δυαδικό σε δεκαδικό

Η διαδικασία της μετατροπής από δυαδικό σε δεκαδικό είναι κάπως πιο απλή. Η διαδικασία λοιπόν είναι ως εξής :

1. Μετατρέπουμε το σημαντικό μέρος στο δεκαδικό
2. Μετατρέπουμε τον εκθέτη στο δεκαδικό
3. Αφαιρούμε την τιμή πόλωσης από τον εκθέτη.
4. Υπολογίζουμε τον αριθμό που είναι στην μορφή (1.2).

3.4 Bit συμπλήρωσης (Guard Bits)

Πριν από κάθε πράξη κινητής υποδιαστολής τα σημαντικά μέρη συνήθως στις πράξεις αποθηκεύονται σε καταχωρητές μεγαλύτερου μήκους. Αυτά τα επιπλέον bit ονομάζονται guard bits. Το μήκος του καταχωρητή είναι πάντοτε μεγαλύτερο από το μήκος του σημαντικού μέρους συν το επιπλέον κρυφό bit.

Ο λόγος για την συμπλήρωση του σημαντικού μέρους είναι ότι όταν κάνουμε μετατοπίσεις δεξιά και αριστερά του σημαντικού μέρους για να μην χαθούν κάποια bit. Ειδικά όταν έχουμε ακραίες τιμές μπορεί να χαθούν κάποια bit και να έχουμε αποτέλεσμα 0. Για παράδειγμα όταν έχουμε έναν αριθμό με μόνο το τελευταίο bit του να είναι 1 τότε στην μετατόπιση του στα αριστερά έστω και μια φορά αυτό το bit θα χαθεί και θα έχουμε mantissa με μηδέν. Πράγμα το οποίο δεν είναι σωστό.

3.5 Στρογγυλοποίηση

Ίσως είναι η καλύτερη στιγμή να περιγράψουμε την στρογγυλοποίηση. Η στρογγυλοποίηση επηρεάζει την ακρίβεια του αποτελέσματος. Αναφέραμε ότι τα σημαντικά μέρη συνήθως στης πράξεις αποθηκεύονται σε καταχωρητές μεγαλύτερου μήκους. Όταν όμως το αποτέλεσμα επανατίθεται στη μορφή της κινητής υποδιαστολής τα επιπλέον bit πρέπει να απομακρυνθούν.

Υπάρχουν κάποιες τεχνικές στρογγυλοποίησης. Σύμφωνα όμως με το πρότυπο IEEE υπάρχουν 4 τύποι στρογγυλοποίησης.

1) Στρογγυλοποίηση προς το πλησιέστερο,
2) Στρογγυλοποίηση προς το συν άπειρο,
3) Στρογγυλοποίηση προς το μείον άπειρο,
4) Στρογγυλοποίηση προς το μηδέν.

Ας δούμε τώρα τους 4 τρόπους στρογγυλοποίησης ξεχωριστά.

3.5.1 Στρογγυλοποίηση προς το πλησιέστερο

Το αποτέλεσμα στρογγυλοποιείται προς τον πλησιέστερο δυαδικό αριθμό ο οποίος μπορεί να αναπαρασταθεί. Αυτός είναι ο εξορισμού τρόπος στρογγυλοποίησης που αναφέρεται στο πρότυπο. Ορίζεται ως εξής:

Η τιμή που μπορεί να αναπαρασταθεί πλησιέστερα στο απείρως ακριβές αποτέλεσμα, είναι εκείνη που θα εμφανιστεί. Αν οι δυο πλησιέστερες τιμές είναι εξίσου κοντά, τότε θα εμφανιστεί εκείνη που έχει ως λιγότερο ψηφίο το 0.

Για παράδειγμα στην απλή μορφή του single precision που χρησιμοποιούμε αν τα επιπλέον bit πέραν των 23 στο σημαντικό μέρος είναι 10100 τότε κάνουμε στρογγυλοποίηση προσθέτοντας έναν άσο στο τελευταίο bit του σημαντικού μέρους. Εάν όμως τα επιπλέον bit είναι 00111 τότε απλώς κόβουμε τα επιπλέον bit. Στην ουσία δηλαδή εάν τα επιπλέον bit είναι παραπάνω από το μισό του τελευταίου bit

που μπορεί να αναπαρασταθεί τότε κάνουμε στρογγυλοποίηση προσθέτοντας έναν άσο στο τελευταίο bit του σημαντικού μέρους.

Επίσης στο πρότυπο αναφέρεται και η ειδική περίπτωση που τα επιπλέον bit είναι ακριβώς στην μέση του τελευταίου bit που μπορεί να αναπαρασταθεί. Θα μπορούσαμε να κόψουμε για πάντα αυτά τα bit. Δηλαδή πάντοτε να κάνουμε αποκοπή των επιπλέον bit. Όμως με αυτό τον τρόπο εισάγουμε μια μικρή συσσωρευτική τιμή πόλωσης κατά την εκτέλεση μιας ακολουθίας υπολογισμών. Αυτό που θέλουμε στην ουσία είναι μια μέθοδος χωρίς πόλωση. Μια δυνατή προσέγγιση θα ήταν να στρογγυλοποιήσουμε προς τα πάνω η προς τα κάτω βάση με έναν τυχαίο αριθμό. Ώστε κατά μέσο όρο το αποτέλεσμα θα ήταν χωρίς πόλωση. Το επιχείρημα εναντίον αυτής της προσέγγισης αυτής όμως είναι ότι δεν μας παράγει ακριβή αποτελέσματα. Η προσέγγιση που υιοθετείται στο πρότυπο IEEE είναι να εξαναγκάζεται το αποτέλεσμα να είναι άρτιο. Αν το αποτέλεσμα ενός υπολογισμού είναι ακριβώς στο μέσο μεταξύ 2 αριθμών οι οποίοι μπορούν να αναπαρασταθούν, η τιμή στρογγυλοποιείται αν το τελευταίο bit του σημαντικού μέρους είναι άσος “1” τότε στρογγυλοποιείται προς τα πάνω. Αλλιώς τα επιπλέον bit απομακρύνονται χωρίς να γίνει καμία στρογγυλοποίηση.

3.5.2 Στρογγυλοποίηση προς το συν άπειρο

Αυτή η τεχνική είναι χρήσιμη στην αριθμητική των διαστημάτων. Το αποτέλεσμα της πράξης στρογγυλοποιείται προς το συν άπειρο. Εάν το πρόσημο είναι θετικό και το επιπλέον είναι διάφορο του μηδέν δηλαδή διάφορο του 00000 τότε το αποτέλεσμα στρογγυλοποιείται προσθέτοντας έναν άσο στο τέλος του σημαντικού μέρους.

3.5.3 Στρογγυλοποίηση προς το μείον άπειρο

Αυτή η τεχνική είναι παρόμοια με την τεχνική της στρογγυλοποίησης προς το συν άπειρο. Εάν δηλαδή τα επιπλέον bit είναι διάφορο του μηδέν αλλά το πρόσημο αυτή τη φορά είναι αρνητικό τότε στρογγυλοποιείται προσθέτοντας έναν άσο στο τέλος του σημαντικού μέρους.

3.5.4 Στρογγυλοποίηση προς το μηδέν

Η στρογγυλοποίηση προς το μηδέν είναι στην ουσία είναι η απλή τεχνική της αποκοπής των επιπλέον bit. Εδώ όμως το μέτρο της τιμής που έχει υποστεί περικοπή είναι πάντοτε μικρότερο η ίσο με την πιο ακριβή αρχική τιμή. Στην διαδικασία αυτή εισάγεται μια τιμή πόλωσης προς το μηδέν πάντοτε.

Για παράδειγμα ο αριθμός 1,766 με την περικοπή θα γίνει : 1,7659999.

3.6 Το διάστημα των αριθμών που μπορεί να αναπαρασταθεί σε μία λέξη 32 bit

Με τη μορφή κινητής υποδιαστολής είναι δυνατή η αναπαράσταση των εξής διαστημάτων :

- Αρνητικοί αριθμοί : μεταξύ $-(2-2^{23}) \times 2^{128}$ και -2^{-127}
- Θετικοί αριθμοί : μεταξύ 2^{-127} και $(2-2^{-23}) \times 2^{128}$

3.7 Διαστήματα που δεν περιλαμβάνονται στην αναπαράσταση

5 διαστήματα από την ευθεία των αριθμών δεν περιλαμβάνονται στα διαστήματα αυτά :

1. Αρνητική υπερχείλιση

   Αρνητικοί αριθμοί μικρότεροι από $-(2-2^{23}) \times 2^{128}$

2. Αρνητική υποχείλιση

   Αρνητικοί αριθμοί μεγαλύτερη από -2^{-127}

3. Το μηδέν

   Επίσης και το μηδέν δεν περιλαμβάνεται

4. Θετική υποχείλιση

   Θετικοί αριθμοί μικρότεροι από 2^{-127}

5. Θετική υπερχείλιση

   Θετικοί αριθμοί μεγαλύτερη από $(2-2^{-23}) \times 2^{128}$.

Η αναπαράσταση, όπως παρουσιάστηκε, δεν μπορεί να αναπαραστήσει τιμή ίση με 0. Όμως όπως θα δούμε οι αναπαραστάσεις κινητής υποδιαστολής περιλαμβάνουν μια ειδική ακολουθία bit για την αναπαράσταση του μηδενός.

3.7.1 Η υπερχείλιση

Υπερχείλιση εκθέτη

Ένας θετικός εκθέτης υπερβαίνει την μέγιστη δυνατή τιμή εκθέτη. Σε μερικά συστήματα, αυτή μπορεί να καθορίζεται ως +∞ ή -∞.

Υπερχείλιση σημαντικού μέρους

Η πρόσθεση δυο σημαντικών μερών του ίδιου πρόσημου μπορεί να έχει σαν αποτέλεσμα σε απομάκρυνση του πιο σημαντικού bit. Αυτό μπορεί να διορθωθεί με επανευθυγράμμιση όπως θα δούμε παρακάτω.

3.7.2 Η υποχείλιση

Υποχείλιση εκθέτη

Ένας αρνητικός εκθέτης είναι μικρότερος από την ελάχιστη δυνατή τιμή εκθέτη (π.χ. -200 που είναι μικρότερη από -127). Αυτό σημαίνει ότι ο αριθμός είναι πολύ μικρός για να αναπαρασταθεί, και μπορεί να αναπαρασταθεί με 0.

Υποχείλιση σημαντικούμέρους

Στη διαδικασία ευθυγράμμισης των σημαντικών μερών, μπορεί να μετακινηθούν ψηφία πέρα από το δεξί άκρο του σημαντικού μέρους. Όπως θα δούμε, απαιτείται κάποιας μορφής στρογγυλοποίηση.

Η υποχείλιση είναι λιγότερο σημαντικό πρόβλημα επειδή το αποτέλεσμα μπορεί γενικά να προσεγγιστεί ικανοποιητικά με το 0.

3.8 Ειδικές τιμές

Δεν ερμηνεύονται με το συνήθη τρόπο όλες οι ακολουθίες bit στις μορφές που καθόρισε το πρότυπο της IEEE. Αντίθετα, μερικές ακολουθίες bit χρησιμοποιούνται για την αναπαράσταση ειδικών τιμών. Το IEEE έχει κάποιες ειδικές τιμές. Οι ακρότατες τιμές εκθέτη [όλα τα bit 1 ή 0 (255 στην απλή μορφή, 2047 στη μορφή διπλής ακρίβειας)] καθορίζουν ειδικές τιμές. Οι παρακάτω πίνακες δείχνουν για απλή και διπλή ακρίβεια της ειδικές τιμές:

ΕΙΔΙΚΗ ΤΙΜΗ	***Πρόσημο***	***Πολωμένος εκθέτης***	***Κλάσμα***	***Τιμή***
ΘΕΤΙΚΟ ΜΗΔΕΝ	0	0	0	0
ΑΡΝΗΤΙΚΟ ΜΗΔΕΝ	1	0	0	-0
ΘΕΤΙΚΟ ΑΠΕΙΡΟ	0	255 Όλα 1	0	∞
ΑΡΝΗΤΙΚΟ ΑΠΕΙΡΟ	1	255 Όλα 1	0	-∞
ΣΙΩΠΗΡΟ NaN	0 ή 1	255 Όλα 1	≠ 0	NaN
NaN ΣΗΜΑΤΟΔΟΣΙΑΣ	0 ή 1	255 Όλα 1	≠ 0	NaN
ΘΕΤΙΚΟΣ	0	0<e<255	f	$2^{e-127}(1.f)$
ΜΗΔΕΝΙΚΟΣ				
ΑΡΝΗΤΙΚΟΣ	1	0<e<255	f	$-2^{e-127}(1.f)$
ΜΗΔΕΝΙΚΟΣ				
ΘΕΤΙΚΟΣ	0	0	f≠0	$2^{e-126}(0.f)$
ΑΠΟΚΑΝΟΝΙΚΟΠΟΙΗΜΕΝΟΣ				
ΑΡΝΗΤΙΚΟΣ	1	0	f≠0	-2e-126(0.f)
ΑΠΟΚΑΝΟΝΙΚΟΠΟΙΗΜΕΝΟΣ				

ΣΧΗΜΑ 6. Ειδικές τιμές για απλή ακρίβεια

	ΔΙΠΛΗ ΑΚΡΙΒΕΙΑ (64 bit)			
ΕΙΔΙΚΗ ΤΙΜΗ	***Πρόσημο***	***Πολωμένος εκθέτης***	***Κλάσμα***	***Τιμή***
ΘΕΤΙΚΟ ΜΗΔΕΝ	0	0	0	0
ΑΡΝΗΤΙΚΟ ΜΗΔΕΝ	1	0	0	-0
ΘΕΤΙΚΟ ΑΠΕΙΡΟ	0	2047 Όλα 1	0	∞
ΑΡΝΗΤΙΚΟ ΑΠΕΙΡΟ	1	2047 Όλα 1	0	-∞
ΣΙΩΠΗΡΟ NaN	0 ή 1	2047 Όλα 1	≠ 0	NαN
NaN ΣΗΜΑΤΟΔΟΣΙΑΣ	0 ή 1	2047 Όλα 1	≠ 0	NaN
ΘΕΤΙΚΟΣ	0	0<e<2047	f	$2^{e-1023}(1.f)$
ΜΗΔΕΝΙΚΟΣ				
ΑΡΝΗΤΙΚΟΣ	1	0<e<2047	f	$-2^{e-1023}(1.f)$
ΜΗΔΕΝΙΚΟΣ				
ΘΕΤΙΚΟΣ	0	0	f≠0	$2^{e-1022}(0.f)$
ΑΠΟΚΑΝΟΝΙΚΟΠΟΙΗΜΕΝΟΣ				
ΑΡΝΗΤΙΚΟΣ	1	0	f≠0	$-2^{e-1022}(0.f)$
ΑΠΟΚΑΝΟΝΙΚΟΠΟΙΗΜΕΝΟΣ				

ΣΧΗΜΑ 7. Ειδικές τιμές για διπλή ακρίβεια

3.8.1 Κανονικοποιημένοι μη μηδενικοί αριθμοί

Για τιμές εκθέτη στο διάστημα από 1 έως 254 για απλή ακρίβεια, και 1 έως 2046 για διπλή ακρίβεια, αναπαρίστανται κανονικοποιημένοι μη μηδενικοί αριθμοί κινητής υποδιαστολής. Ο εκθέτης είναι πολωμένος, ώστε ο αριθμός των εκθετών είναι από -126 έως +127 για απλή ακρίβεια και από -1022 έως +1023. Ένας κανονικοποιημένος αριθμός απαιτεί ένα bit ίσο με 1 στα αριστερά της δυαδικής υποδιαστολής. Αυτό το bit υπονοείται, δίνοντας ένα ενεργό εύρος σημαντικού μέρους με 24 ή 53 bit (το οποίο ονομάζεται κλάσμα στο πρότυπο).

πχ. 0 01001010 10100000000000000000000

3.8.2 Θετικό ή αρνητικό μηδέν

Ένας εκθέτης ίσος με μηδέν, μαζί με κλάσμα ίσο με μηδέν ,αναπαριστά θετικό ή αρνητικό μηδέν, ανάλογα με το bit πρόσημου. Είναι χρήσιμο να μπορούμε να αναπαραστήσουμε μια τιμή ακριβώς ίση μηδέν.

3.8.3 Θετικό ή αρνητικό άπειρο

Ένας εκθέτης με όλα του ψηφία ίσα με 1 μαζί με ένα μηδενικό κλάσμα αναπαριστά θετικό ή αρνητικό άπειρο ανάλογα με το bit πρόσημου. Είναι χρήσιμο να έχουμε μία αναπαράσταση του απείρου. Αυτό επιτρέπει στο χρήστη να αποφασίσει αν θα συμπεριφερθεί στην υπερχείλιση ως συνθήκη σφάλματος ή θα μεταφέρει την τιμή ∞ και θα προχωρήσει με την εκτέλεση του προγράμματος.

3.8.4 Αποκανονικοποιημένοι αριθμοί

Ένας εκθέτης με όλα τα ψηφία του 0 μαζί με ένα μη μηδενικό κλάσμα αναπαριστά ένα αποκανονικοποιημένο αριθμό. Στην περίπτωση αυτή, το bit στα αριστερά της δυαδικής υποδιαστολής είναι μηδέν, και ο πραγματικός εκθέτης είναι -126 ή -1022. Ο αριθμός είναι θετικός ή αρνητικός ανάλογα με το bit πρόσημου.

πχ. 0 00000000 10100000000000000000000
(αποκανονικοποιημένος αριθμός)

Οι αποκανονικοποιημένοι αριθμοί περιλαμβάνονται στο πρότυπο για να αντιμετωπίσουν καταστάσεις υποχείλισης εκθέτη. Όταν ο εκθέτης του αποτελέσματος γίνεται πολύ μικρής τιμής (ένας αρνητικός εκθέτης με υπερβολικά μεγάλο μέτρο) το αποτέλεσμα αποκανονικοποιείται με δεξιά μετατόπιση του σημαντικού μέρους και αύξηση του εκθέτη μέχρι ένα σημείο που μπορεί να αναπαρασταθεί.

Το παρακάτω σχήμα παρουσιάζει την μορφή single, χωρίς και με αποκανονικοποιημένους αριθμούς. Οι αριθμοί που μπορούν να αναπαρασταθούν μπορούν να ομαδοποιηθούν σε διαστήματα της μορφής $[2^n, 2^{n+1}]$. Μέσα σε κάθε τέτοιο διάστημα, το τμήμα του εκθέτη στον αριθμό παραμένει σταθερό, ενώ το κλάσμα μεταβάλλεται, παράγοντας μια ομοιόμορφη κατανομή των αριθμών που μπορούν να αναπαρασταθούν μέσα στο διάστημα.

Καθώς πλησιάζουμε πιο κοντά στο 0, κάθε διαδοχικό διάστημα έχει το μισό εύρος από το προηγούμενο διάστημα, αλλά, περιέχει το ίδιο πλήθος αριθμών οι οποίοι μπορούν να αναπαρασταθούν. Έτσι η πυκνότητα των αριθμών που μπορούν να αναπαρασταθούν αυξάνεται καθώς πλησιάζουμε στο μηδέν. Όμως αν χρησιμοποιηθούν μόνο κανονικοποιημένοι αριθμοί, υπάρχει ένα διάκενο ανάμεσα στο μηδέν με το μικρότερο κανονικοποιημένο αριθμό.

Στην περίπτωση με 32 bit, υπάρχουν 2^{32}αριθμοί που μπορούν να αναπαρασταθούν σε κάθε διάστημα, και μικρότερος θετικός είναι ο 2^{-126}. Με την προσθήκη και των αποκανονικοποιημένων, προστίθενται επιπλέον 2^{23}αριθμοίομοιόμορφαμεταξύ 0 και του 2^{-126}.

3.8.5 NaN (Not a Number)

Ένας εκθέτης με όλα του τα ψηφία ίσα με 1, μαζί με ένα μη μηδενικό κλάσμα παίρνει την τιμή NaN, που σημαίνει όχι αριθμός, και χρησιμοποιείται για να σηματοδοτήσει διάφορες συνθήκες εξαίρεσης. Υπάρχουν 2 είδη NaN’s σηματοδοτικά και σιωπηρά NaN’s.

3.8.5.1 Σηματοδοτικά NaN’s

Μία NaN είναι μια συμβολική οντότητα που είναι κωδικοποιημένη σε μορφή κινητής υποδιαστολής. Μια σηματοδοτική NaN σηματοδοτεί μία εξαίρεση άκυρης πράξης, όποτε εμφανίζεται ως τελεστής. Οι σηματοδοτικές NaN παρέχουν τιμές για μεταβλητές χωρίς αρχικές τιμές και παρα αριθμητικές βελτιώσεις, οι οποίες δεν είναι αντικείμενο του προτύπου.

3.8.5.2 Σιωπηρά NaN's

Μια σιωπηρή NaN μεταδίδεται διαμέσου σχεδόν κάθε αριθμητικής πράξης χωρίς να σηματοδοτείται μία εξαίρεση. Και οι δυο τύποι NaN έχουν την ίδια μορφή.

3.8.6 Αόριστη Μορφή

Πρέπει να αναφέρουμε ότι στο πρότυπο ορίζεται και μορφή η λεγόμενη αόριστη μορφή. Σύμφωνα με το πρότυπο η αόριστη μορφή έχει πρόσημο αρνητικό δηλαδή 1. Ο εκθέτης αποτελείται από άσους. Ενώ στο σημαντικό μέρος μόνο το πρώτο bit είναι 1 και όλα τα υπόλοιπα μηδέν.

Το πρότυπο έχει ορίσει για κάθε μορφή κάποια ακρίβεια ψηφίων. Για το τύπο απλής μορφής η ακρίβεια των δεκαδικών ψηφίων είναι 8. Άρα δηλαδή τα επιπλέον δεκαδικά ψηφία κόβονται μετά την στρογγυλοποίηση. Για παράδειγμα ο αριθμός 1,2955 x 10^{13}μετατρέπεται στο δυαδικό ως εξής με στρογγυλοποίηση προς το πλησιέστερο:

0-10101010-1.01111001000010100100011

Εάν τώρα όμως θέλουμε να μετατρέψουμε τον αριθμό αυτό πάλι σε δεκαδικό έχουμε αποτέλεσμα ίσο με:

1,2955000152317231104 x 10^{13}

Παρατηρήστε ότι ο αριθμός είναι διαφορετικός από τον αρχικό. Εδώ λοιπόν παίρνει μέρος η ακρίβεια ψηφίων και παίρνουμε μόνο τα πρώτα 8 δεκαδικά ψηφία υπόψη και έχουμε τελικά :

1,2955000 x 10^{13}

ΚΕΦΑΛΑΙΟ 4^{ο} :

Αριθμητική αριθμών κινητής υποδιαστολής. Πρόσθεση, Αφαίρεση, Πολλαπλασιασμός και Διαίρεση

4.1 Πρόσθεση αριθμών κινητής υποδιαστολής

Όσο στο δεκαδικό σύστημα η αριθμητική κινητής υποδιαστολής είναι εύκολη, στην δυαδική αριθμητική κινητής υποδιαστολής η πρόσθεση και η αφαίρεση είναι πιο σύνθετες πράξεις από τον πολλαπλασιασμό και την διαίρεση.

Υπάρχουν 4 βασικές φάσεις στον αλγόριθμο για την πρόσθεση:

1) Έλεγχος για μηδέν
2) Ευθυγράμμιση εκθετών
3) Πρόσθεση των σημαντικών μερών
4) Κανονικοποίηση του αποτελέσματος.

Ο αλγόριθμος της πρόσθεσης είναι όπως αυτό του σχήματος 11.

Θα κάνουμε μια επεξήγηση του αλγορίθμου βήμα βήμα ώστε να γίνει πιο κατανοητό. Πρώτα απ' όλα πρέπει να τονίσουμε ότι το κρυφό bit όπως και στης άλλες πράξεις και στην πρόσθεση πρέπει να εμφανιστεί, αλλιώς το αποτέλεσμα είναι λάθος. Επίσης το πρόσημο πρέπει να φαίνεται. Τελικά **για να κάνουμε σωστά την πράξη** πρέπει να έχουμε το σημαντικό μέρος με το πρόσημο, το bit στα αριστερά της υποδιαστολής και το υπόλοιπο κομμάτι του σημαντικού μέρους. Το πρόσημο όμως το παίρνουμε υπόψη μας μόνο όταν κάνουμε την πρόσθεση η αφαίρεση. Ενώ στο πολλαπλασιασμό και την διαίρεση το πρόσημο υπολογίζεται ξεχωριστά όπως θα δούμε πιο κάτω.

Θεωρούμε δυο αριθμούς κινητής υποδιαστολής A και B και το αποτέλεσμα μας θα αποθηκεύεται στο C.

Πρώτα ελέγχουμε εάν το A είναι μηδέν και εάν είναι μηδέν το αποτέλεσμα ισούται με το B. Εάν λοιπόν το A είναι διάφορο του μηδενός, ελέγχουμε αν το B είναι μηδέν . Αν το B είναι μηδέν το αποτέλεσμα είναι με το A αλλιώς συνεχίζουμε με την επόμενη φάση του αλγορίθμου.

Αν λοιπόν δεν **είναι** μηδέν το A ή το B, στη συνέχεια, όπως και στο δεκαδικό σύστημα**,** κάνουμε ευθυγράμμιση εκθετών. Επειδή η μετατόπιση αριστερά του

αριθμού είναι αδύνατη διότι χάνουμε τα σημαντικά bit, κάνουμε μετατόπιση προς τα δεξιά. Έτσι για την ευθυγράμμιση πρέπει να επιλέξουμε τον εκθέτη που είναι μικρότερος. Στη συνέχεια κάνουμε μετατόπιση του σημαντικού μέρους προς τα δεξιά αυξάνοντας τον εκθέτη. Κάθε φορά που αυξάνουμε πρέπει να ελέγχουμε εάν το σημαντικό μέρος είναι μηδέν. Εάν είναι μηδέν τότε το αποτέλεσμα ισούται με το άλλον αριθμό που έχει το μεγαλύτερο εκθέτη. Αυτή είναι μία περίπτωση που συμβαίνει όταν έχουμε 2 αριθμούς με μεγάλη διαφορά εκθετών. Για παράδειγμα $1,25 \times 2^{30} + 1,25$ ισούται με το πρώτο αριθμό και μας δίνει αποτέλεσμα $1,25 \times 2^{30}$.

Μόλις λοιπόν κάνουμε ευθυγράμμιση εκθετών ερχόμαστε στο στάδιο πρόσθεσης των σημαντικών μερών . Όμως εδώ πρέπει να διευκρινίσουμε ότι εάν υπάρχει αριθμός με αρνητικό πρέπει να γίνει συμπλήρωμα ως προς το 2 για λόγους ευκολίας. Το συμπλήρωμα ως προς 2 θα το περιγράψουμε στην αμέσως επόμενη παράγραφο. Κάνουμε λοιπόν πρόσθεση των 2 σημαντικών μερών με τον κλασικό τρόπο που ξέρουμε.

Στη συνέχεια ελέγχουμε εάν έχουμε υπερχείλιση σημαντικού μέρους. Σε αυτή την περίπτωση μετατοπίζουμε δεξιά το αποτέλεσμα και αυξάνουμε τον εκθέτη του αποτελέσματος.

Εδώ λοιπόν εάν το αποτέλεσμα έχει πρόσημο αρνητικό κάνουμε συμπλήρωμα ως προς το 2 του αποτελέσματος ώστε να πάρουμε την πραγματική του τιμή.

Στη συνέχεια λοιπόν ελέγχουμε εάν έχουμε υπερχείλιση του εκθέτη μας έχει υπερχείλιση. Σταματάμε την πρόσθεση με μία αναφορά υπερχείλισης.

Στη συνέχεια εάν το αποτέλεσμα δεν είναι κανονικοποιημένο, κάνουμε κανονικοποίηση του αποτελέσματος. Στην περίπτωση της κανονικοποίησης (*βλ. **2.5***) όταν κάνουμε μετατόπιση αριστερά το σημαντικό μέρος πρέπει να ελέγξουμε εάν υπάρχει υποχείλιση εκθέτη.

Εάν το αποτέλεσμα έχει κανονικοποιηθεί σωστά στο τέλος κάνουμε στρογγυλοποίηση(*βλ2.5*).

Ας μελετήσουμε τώρα κάποια μέρη της πρόσθεσης που θα μας χρειαστούν και στη συνέχεια :

4.2 Ευθυγράμμιση εκθετών

Η διαδικασία της ευθυγράμμισης είναι μία πολύ απλή διαδικασία. Πρώτα αυξάνουμε τον εκθέτη, στη συνέχεια μετατοπίζουμε το σημαντικό μέρος προς τα δεξιά και ελέγχουμε εάν το σημαντικό μέρος είναι μηδέν. Εάν όχι, κάνουμε ευθυγράμμιση μέχρι να γίνουν οι 2 εκθέτες ίσοι. Ας θεωρήσουμε για παράδειγμα 2 αριθμούς και να τους κάνουμε ευθυγράμμιση:

$$A=1,0100....000 \times 2^{3}$$

$$B=1,1110....000 \times 2^{-2}$$

Όπως βλέπετε ο αριθμός Β έχει εκθέτη μικρότερο. Άρα όπως αναφέραμε πιο πριν ευθυγραμμίζουμε αυξάνοντας τον εκθέτη του Β και μετατοπίζοντας τον αριθμό προς τα δεξιά και έχουμε:

Αρχικό νούμερο : $1,1110....000 \times 2^{-2}$

Μετατόπιση και αύξηση εκθέτη: $0,11110....000 \times 2^{-1}$ Μετατόπιση και αύξηση εκθέτη: $0,011110....000 \times 2^{0}$ Μετατόπιση και αύξηση εκθέτη: $0,0011110....000 \times 2^{1}$ Μετατόπιση και αύξηση εκθέτη: $0,00011110....000 \times 2^{2}$

Τελικά λοιπόν θα έχουμε τον αριθμό: $0,0001111....000 \times 2^{2}$. Όταν κάνουμε αυτή την διαδικασία είναι πολύ σημαντικό να προσέχουμε να μην γίνει το σημαντικό μέρος 0. Διότι αυτό είναι ένα από τα κριτήρια τερματισμού της πράξης.

4.3 Υπερχείλιση σημαντικού μέρους

Είναι ένα θέμα σημαντικό στην διαδικασία πρόσθεσης των σημαντικών μερών. Υπερχείλιση σημαντικού μέρους προκύπτει όταν τα πρόσημά των 2 αριθμών είναι ίδια. Θα πρέπει να προσέχουμε όμως διότι η ισότητα των εκθετών ισχύει στην φάση πρόσθεσης η αφαίρεσης των σημαντικών μερών. Ενώ στην αφαίρεση όταν η εκθέτες είναι διαφορετικοί τότε μας προκύπτει υπερχείλιση σημαντικού μέρους. Για παράδειγμα εάν προσθέσουμε τους αριθμούς:

$$\begin{array}{r} 1.1100...00 \\ +\ 1.0100...00 \\ \hline 11.0000...00 \end{array}$$

Ο αριθμός έχει υπερχείλιση σημαντικού μέρους δηλαδή τα bit στα αριστερά της υποδιαστολής είναι πάνω από 1 οπότε κάνουμε μετατόπιση δεξιά και αυξάνουμε τον εκθέτη του αποτελέσματος.

4.4 Υπερχείλιση εκθέτη

Όταν κάνουμε πρόσθεση εκθετών στον πολλαπλασιασμό ή αφαίρεση εκθετών στην διαίρεση μπορεί να προκύψει υπερχείλιση εκθέτη. Ενώ στην πρόσθεση και αφαίρεση, η υπερχείλιση εκθέτη μπορεί να προκύψει όταν έχουμε υπερχείλιση σημαντικού μέρους και χρειαστεί να αυξήσουμε την τον εκθέτη.

Ας δούμε ένα παραδείγματα υπερχείλισης εκθέτη στην πρόσθεση:

1) Ας υποθέσουμε ότι έχουμε να κάνουμε πρόσθεση 2 αριθμούς με εκθέτη 11111110 σε single μορφή. Τα πρόσημα τους είναι θετικά. Ενώ τα σημαντικά μέρη τους είναι:

A) 1.0110…..00

B) 1.1001…..00

Αφού λοιπόν κάνουμε την πρόσθεση αυτών των αριθμών έχουμε αποτέλεσμα: 10.1111…..00. Όπως παρατηρείτε έχουμε υπερχείλιση σημαντικού μέρος. Χρειάζεται να κάνουμε μετατόπιση αριστερά το σημαντικό μέρος και να αυξήσουμε τον εκθέτη του αποτελέσματος. Μετά την μετατόπιση του σημαντικού μέρους και αύξηση του εκθέτη έχουμε το εξής αποτέλεσμα:

C) 1.01111……00

Και θα έχουμε εκθέτη μετά την αύξηση του: 11111111. Αυτός ο εκθέτης όπως βλέπετε έχει υπερχείλιση. Σε αυτή την περίπτωση εισάγουμε στον καταχωρητή την τιμή του απείρου, + άπειρο η – άπειρο.

4.5 Υποχείλιση εκθέτη

Η υποχείλιση εκθέτη μπορεί να συμβαίνει όταν κάνουμε κανονικοποίηση στις τέσσερις πράξεις και όταν κάνουμε πρόσθεση και αφαίρεση εκθετών αντίστοιχα στον πολλαπλασιασμό και την διαίρεση. Στην δεύτερη μπορεί να συμβεί υποχείλιση μόνο όταν κάνουμε αφαίρεση εκθετών και ο δεύτερος εκθέτης είναι μεγαλύτερος από τον πρώτο.

Για να μας γίνει πιο κατανοητό ας δούμε ένα παράδειγμα που θα συμβεί υποχείλιση. Ας υποθέσουμε ότι θέλουμε να κάνουμε διαίρεση και έχουμε τους εξής δύο εκθέτες: 00000001 και 00000100. Στην διαίρεση η εκθέτες αφαιρούνται. Κάνουμε λοιπόν αφαίρεση των εκθετών και έχουμε αποτέλεσμα: 11111101. Αυτό είναι έναν αριθμό μεγαλύτερο από τον πρώτο αριθμό. Πράγμα το οποίο σημαίνει ότι έχουμε υποχείλιση εκθέτη. Σε αυτή την περίπτωση βάζουμε στον καταχωρητή την τιμή της υποχείλισης.

4.6 Συμπλήρωμα ως προς 2

Για να βρούμε την πραγματική τιμή ενός αρνητικού αριθμού πρέπει να κάνουμε πάλι την αντίστροφη διαδικασία. Δηλαδή αντιστρέφουμε όλα τα bit και προσθέτουμε έναν άσο στο τελευταίο bit χωρίς να πειράξουμε το bit πρόσημου.

4.7 Μερικά παραδείγματα πρόσθεσης

Παράδειγμα 1:

Ας υποθέσουμε ότι έχουμε τους παρακάτω αριθμούς Α και Β :

0	10100000	1.01110100100001110110111	01000

0	10011010	1.10101101001001110100100	00000

1) Πρώτα ελέγχουμε εάν ένα από τα δυο νούμερα είναι μηδέν. Στην περίπτωσή μας κανένα από τα δύο δεν είναι μηδέν.

2) Στη συνέχεια βλέπουμε ότι οι εκθέτες δεν είναι ίσοι. Κάνουμε ευθυγράμμιση εκθετών αυξάνοντας τον μικρότερο εκθέτη. Ο αριθμός Β ευθυγραμμίζεται όπως είχαμε αναφέρει στην ενότητα 3.2. και έχουμε τελικά τους εξής αριθμούς:

0	10100000	1.01110100100001110110111	01000

0	10100000	0.00000110101101001001110	10010

3) Ελέγχουμε εάν το σημαντικό μέρος είναι μηδέν κάθε φορά που κάνουμε μετατόπιση. Αφού λοιπόν δεν έχουμε, κάνουμε πρόσθεση των σημαντικών μερών παίρνοντας υπόψη τα πρόσημα των αριθμών. Και έχουμε το εξής αποτέλεσμα μετά την πρόσθεση :

	0--	1.01110100100001110110111--	01000
	0--	0.00000110101101001001110--	10010
+			
	0--	1.01111011001111000000101--	11010

4) Στη συνέχεια ελέγχουμε εάν το σημαντικό μέρος είναι 0. Βλέπουμε ότι δεν είναι και προχωράμε στο επόμενο βήμα.

5) Φτάσαμε λοιπόν στο στάδιο της κανονικοποίησης. Εδώ βλέπουμε ότι το αποτέλεσμα είναι κανονικοποιημένο και δεν χρειάζεται να κάνουμε κανονικοποίηση.

6) Στο τελικό στάδιο κάνουμε στρογγυλοποίηση. Εδώ λοιπόν μπορούμε να επιλέξουμε ένα από τους 4 τρόπους στρογγυλοποίησης. Εμείς θα προτιμήσουμε την στρογγυλοποίηση προς το πλησιέστερο.

Παράδειγμα 2:

Ας δούμε λοιπόν ένα παράδειγμα στον οποίο έχουμε πρόσημο αρνητικό. Υποθέστε ότι έχουμε τους παρακάτω αριθμούς :

1	01111111	1.11000000000000000000000	00000

0	01111111	1.10000000000000000000000	00000

Όπως βλέπετε για λόγους απλότητας έχουμε βάλει ίδιους εκθέτες. Άρα δεν χρειαζόμαστε ευθυγράμμιση εκθετών. Αυτό που μας μένει είναι να κάνουμε πρόσθεση των σημαντικών μερών. Για να κάνουμε πρόσθεση των σημαντικών μερών πρέπει πριν την πρόσθεση να **βρούμε το** συμπλήρωμα ως προς 2 των αριθμών που έχουν αρνητικό πρόσημο. Όπως βλέπετε ο αριθμός Α έχει πρόσημο αρνητικό οπότε πρέπει να γίνει συμπλήρωμα ως προς 2. Κάνοντας λοιπόν συμπλήρωμα το 2 έχουμε το εξής παρακάτω αριθμό ως Α, που είναι το συμπλήρωμα ως προς 2 του:

1	01111111	0.01000000000000000000000	00000

Ύστερα κάνουμε πρόσθεση των 2 αριθμών και έχουμε το παρακάτω αποτέλεσμα :

1-0.01000000000000000000000-00000

\+ 0-1.10000000000000000000000-00000

1-1.11000000000000000000000-00000

Παίρνουμε ένα αποτέλεσμα αρνητικό. Άρα το αποτέλεσμα μας είναι αρνητικό. Αλλά ο αριθμός μας στην ουσία δεν είναι αυτό. Επειδή είναι αρνητικός ο αριθμός πρέπει να γίνει πάλι συμπλήρωμα ως προς 2 για να βρούμε το πραγματικό αποτέλεσμα. Συμπληρώνοντας λοιπόν ως προς 2 έχουμε αποτέλεσμα:

1-0.01000000000000000000000-00000

Εδώ λοιπόν πρέπει να κάνουμε κανονικοποίηση του αποτελέσματος διότι το δυαδικό ψηφίο στα αριστερά της υποδιαστολής είναι 0.

Κάνοντας λοιπόν κανονικοποίηση έχουμε:

1	01111101	1.00000000000000000000000	00000

Στο τελευταίο στάδιο του αλγόριθμου βλέπουμε πως δεν χρειάζεται στρογγυλοποίηση.

Στην ουσία κάναμε την πράξη -1,75+1,25 και έχουμε αποτέλεσμα -0,50. Άρα στην πρόσθεση και αφαίρεση πρέπει οπωσδήποτε να παίρνουμε υπόψη μας το πρόσημο διότι όταν έχουμε αρνητικό αριθμό παίρνουμε λάθος αποτέλεσμα και

δεύτερον πρέπει να παίρνουμε υπόψη μας ότι εάν το αποτέλεσμα βγει αρνητικό πρέπει οπωσδήποτε πριν κάνουμε οποιαδήποτε πράξη να **βρίσκουμε το** συμπλήρωμα ως προς 2 χωρίς να πειράξουμε τον αρνητικό εκθέτη.

4.8 Αφαίρεση αριθμών κινητής υποδιαστολής

Η αφαίρεση είναι ακριβώς ίδια με την πρόσθεση και μπορεί να υλοποιηθεί με το κύκλωμα της πρόσθεσης. Οι φάσεις του αλγόριθμου είναι ίδιες. Πρώτα όμως εάν θέλουμε να κάνουμε αφαίρεση στην αρχή του αλγόριθμου να αλλάξουμε το πρόσημο του αριθμού Β ώστε να γίνει αφαίρεση. Έτσι χρησιμοποιώντας το κύκλωμα της πρόσθεσης μπορούμε να κάνουμε αφαίρεση.

Οι 4 βασικές φάσεις στον αλγόριθμο για την αφαίρεση είναι όπως της πρόσθεσης:

1) Έλεγχος για μηδέν
2) Ευθυγράμμιση εκθετών
3) Πρόσθεση των σημαντικών μερών
4) Κανονικοποίηση του αποτελέσματος.

Βλέπουμε ότι ο τελικός αριθμός έχει πρόσημο αρνητικό ενώ τα δύο νούμερα είναι θετικά. Αυτό σημαίνει ότι έχουμε υπερχείλιση σημαντικού μέρους. Άρα κάνουμε μετατόπιση αριστερά το σημαντικό μέρος και αυξάνουμε τον εκθέτη και έχουμε τελικό αποτέλεσμα ο οποίος δεν χρειάζεται κανονικοποίηση ή στρογγυλοποίηση :

0	10000000	1.11000000000000000000000	00000

Στην ουσία κάναμε 1,75-(-1,75) ο οποίος μας έδωσε αποτέλεσμα:3,5.

4.9 Πολλαπλασιασμός αριθμών κινητής υποδιαστολής

Ο πολλαπλασιασμός και η διαίρεση κινούμενης υποδιαστολής είναι πολύ απλούστερες διαδικασίες σε σχέση με την πρόσθεση και την αφαίρεση.

Ο αλγόριθμος του πολλαπλασιασμού είναι αυτός που φαίνεται πιο κάτω. Ας εξετάσουμε τον αλγόριθμο βήμα βήμα.

1) Ο πολλαπλασιασμός λοιπόν ξεκινάει ελέγχοντας εάν ο αριθμός είναι Β, τότε το αποτέλεσμα είναι ίσο με το 0.

2) Εάν λοιπόν ο Α δεν είναι μηδέν τότε γίνεται έλεγχος εάν ο αριθμός Β είναι μηδέν. Εάν λοιπόν το Β είναι 0 τότε το αποτέλεσμα είναι ίσο με το 0.

3) Στη συνέχεια στο 3^{ο} στάδιο εάν το Α και το Β δεν είναι 0 γίνεται πρόσθεση των εκθετών.

4) Αμέσως μετά γίνεται αφαίρεση της τιμής πόλωσης. Ο σκοπός για τον οποίο γίνεται αφαίρεση της τιμής πόλωσης είναι επειδή στην πρόσθεση εκθετών και οι 2 εκθέτες είναι πολωμένοι. Άρα αφαιρούμε μια φορά η τιμή πόλωσης ώστε να έχουμε μόνο μια φορά πολωμένο τον εκθέτη.

5) Στο 5^{ο} στάδιο ελέγχουμε εάν έχουμε υπερχείλιση του εκθέτη μετά την πρόσθεση εκθετών. Εάν έχουμε υπερχείλιση εκθέτη τότε το αποτέλεσμα **έχει** υπερχείλιση.

6) Στην αμέσως επόμενη φάση ελέγχουμε εάν έχουμε υποχείλιση του εκθέτη. Εάν έχουμε υποχείλιση εκθέτη τότε το αποτέλεσμα **έχει** υποχείλιση.

7) Εάν λοιπόν δεν έχουμε ούτε υπερχείλιση ούτε υποχείλιση, τότε κάνουμε πολλαπλασιασμό των σημαντικών μερών. Εδώ λοιπόν χρησιμοποιούμε την τεχνική που ξέρουμε στο δημοτικό. Υπάρχουν όμως διάφοροι αλγόριθμοι για τον πολλαπλασιασμό. Εμείς επιλέξαμε αυτή την τεχνική διότι είναι η απλούστερη.

4.10 Διαίρεση αριθμών κινητής υποδιαστολής

Η διαίρεση είναι παρόμοια με τον πολλαπλασιασμό. Τα μοναδικά σημεία που διαφέρουν είναι **πως** αντί να προσθέσουμε τους εκθέτες τους αφαιρούμε. Επίσης όταν ο 2^{ος}αριθμός είναι 0 το αποτέλεσμα είναι άπειρο και όχι 0. Ο αλγόριθμος της διαίρεσης είναι αυτός που φαίνεται παρακάτω.

Ας εξετάσουμε τον αλγόριθμο της διαίρεσης βήμα βήμα.

1) Πρώτα ελέγχουμε εάν το Α είναι 0 τότε το αποτέλεσμα είναι μηδέν.

2) Εάν το Α δεν είναι μηδέν ελέγχουμε εάν ο δεύτερος αριθμός είναι μηδέν. Εάν είναι μηδέν καταχωρούμαι σαν αποτέλεσμα το άπειρο. **Πρόκειται για την** περίπτωση διαίρεσης με το μηδέν.

3) Αφαιρούμε τους εκθέτες.

4) Στη συνέχεια επειδή από τον καινούργιο εκθέτη έχει αφαιρεθεί 2 φορές η τιμή πόλωσης, προσθέτουμε μία φορά την τιμή πόλωσης.

5) Ελέγχουμε εάν έχουμε υπερχείλιση του εκθέτη μετά την αφαίρεση εκθετών. Εάν έχουμε υπερχείλιση εκθέτη τότε το αποτέλεσμα ισούται με άπειρο.

6) Στην αμέσως επόμενη φάση ελέγχουμε εάν έχουμε υποχείλιση του εκθέτη. Εάν έχουμε υποχείλιση εκθέτη τότε το αποτέλεσμα ισούται με υποχείλιση.

7) Εάν λοιπόν δεν έχουμε ούτε υπερχείλιση ούτε υποχείλιση, κάνουμε διαίρεση των σημαντικών μερών. Η διαίρεση των σημαντικών μερών είναι λίγο πολύπλοκη διαδικασία σε σχέση με τον πολλαπλασιασμό. Η διαίρεση γίνεται με συνεχείς μετατοπίσεις και αφαιρέσεις. Ο αλγόριθμος για την διαίρεση των σημαντικών μερών είναι :

μετρητής = μήκος σημαντικού μέρους ΟΣΟ
μετρητής >= 0

ΜΕΤΑΤΟΠΙΣΗ ΑΡΙΣΤΕΡΑ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΕΑΝ Α >= Β

ΤΟΤΕ

A=A-B

LSB ΑΠΟΤΕΛΕΣΜΑΤΟΣ=1

ΑΛΛΙΩΣ

LSB ΑΠΟΤΕΛΕΣΜΑΤΟΣ=0

ΜΕΤΑΤΟΠΙΣΗ ΔΕΞΙΑ Β
μετρητής= μετρητής -1

ΕΠΑΝΑΛΗΨΗ

8) Στη συνέχεια κάνουμε κανονικοποίηση εφόσον χρειάζεται.

9) Τελευταία κάνουμε στρογγυλοποίηση

4.11 Μερικά παραδείγματα διαίρεσης

ΠΑΡΑΔΕΙΓΜΑ 1:

Ας υποθέσουμε ότι έχουμε τους παρακάτω αριθμούς :

0	01111111	1.11000000000000000000000	00000

1	01111111	1.10000000000000000000000	00000

Πρώτα λοιπόν κάνουμε έλεγχο για μηδέν και βλέπουμε ότι δεν έχουμε κανένα μηδέν.

Αφαιρούμε τους εκθέτες και έχουμε :

00000000

Όπως είχαμε αναφέρει και πριν στην αφαίρεση εκθετών, βγαίνει ένα αποτέλεσμα που δεν έχει την τιμή πόλωσης. Προσθέτοντας λοιπόν έχουμε:

01111111

Ελέγχουμε για υποχείλιση και υπερχείλιση εκθέτη αλλά βλέπουμε ότι δεν **ισχύει** κανένα από τα 2. Συνεχίζουμε με την διαίρεση των σημαντικών μερών. Ας κάνουμε λοιπόν βήμα βήμα την διαίρεση των σημαντικών μερών ώστε ο προηγούμενος αλγόριθμος να γίνει πιο κατανοητός. Το αποτέλεσμα καταχωρείται στην C.

Βήμα αρχικοποίησης:

μετρητής = μήκος σημαντικού μέρους (28)

A=1.11000000000000000000000-00000
B=1.10000000000000000000000-00000
C=0.00000000000000000000000-00000

Βήμα 1^{ο} :

A>B άρα αφαιρούμε από το A το B A =
0.10000000000000000000000-00000

Μετατόπιση αριστερά αποτελέσματος LSB
αποτελέσματος=1
C=0.00000000000000000000000-00001

Μετατόπιση δεξιά του B.

B=0.11000000000000000000000-00000
μετρητής=28

Βήμα 2ο:

A>B δεν ισχύει άρα δεν αφαιρούμε

A=0.10000000000000000000000-00000

Μετατόπιση αριστερά αποτελέσματος LSB
αποτελέσματος=0

C=0.00000000000000000000000-00010

Μετατόπιση δεξιά το B.

B=0.01100000000000000000000-00000
μετρητής=27

…….

Την ίδια διαδικασία την συνεχίζουμε μέχρι ο δείκτης να γίνει 0.

Τελικά θα έχουμε ένα αποτέλεσμα :1.01100110011001100110010-10000. Δηλαδή τον αριθμό: -1,3999999 με πρόσημο αρνητικό. Στην ουσία το νούμερο -1,4.

Στη συνέχεια κάνουμε κανονικοποίηση και στρογγυλοποίηση.

ΠΑΡΑΔΕΙΓΜΑ 2:

Ας υποθέσουμε ότι έχουμε τους 2 παρακάτω αριθμούς για διαίρεση :

0	00011011	1.10010101101001011110111	11110

0	11100010	1.11111000110111101111100	01000

Μετά τους ελέγχους για μηδέν βλέπου με ότι κανένα από τα 2 δεν είναι μηδέν. Αφαιρούμε λοιπόν τους εκθέτες και έχουμε:

100111001

Προσθέτουμε λοιπόν την πόλωση και έχουμε :

110111000

Βλέπουμε ότι έχουμε υποχείλιση εκθέτη. Τερματίζουμε την πράξη με αναφορά υποχείλισης.

ΚΕΦΑΛΑΙΟ 5^{ο} :

Η εφαρμογή

5.1 Αριθμομηχανή

Μπαίνοντας ο χρήστης στην εφαρμογή, από προεπιλογή, βρίσκεται στην σελίδα της Αριθμομηχανής. Στο επάνω μέρος, στην πρώτη γραμμή, υπάρχουν τα διαφορετικά αριθμητικά συστήματα που **μπορεί να** χρησιμοποιήσει ο χρήστης. Η δεύτερη γραμμή ανήκει στην «διαγραφή», ενώ στην τρίτη θα βρούμε τα σύμβολα των πράξεων.

Ο χρήστης διαλέγει τον πρώτο του αριθμό και το αριθμητικό σύστημα στο οποίο ανήκει. ΔΕΝ **του επιτρέπεται** να διαλέξει λάθος αριθμητικό σύστημα, όπως π.χ. 985 –>BIN, ενώ δεν **του επιτρέπεται επίσης** να συνεχίσει χωρίς να επιλέξει το κατάλληλο σύστημα.

Ίδια διαδικασία τηρείται και στον δεύτερο, τρίτο κτλ. αριθμό ενώ θα χρειαστεί να πατηθεί το σύμβολο της ισότητας (=) για το τελικό αποτέλεσμα. Στο τέλος, η οθόνη απεικονίζει το αποτέλεσμα, ενώ πάνω από αυτό διακρίνονται με γκρίζο χρώμα οι αριθμοί που χρησιμοποίησε ο χρήστης καθώς και τα αριθμητικά συστήματα.

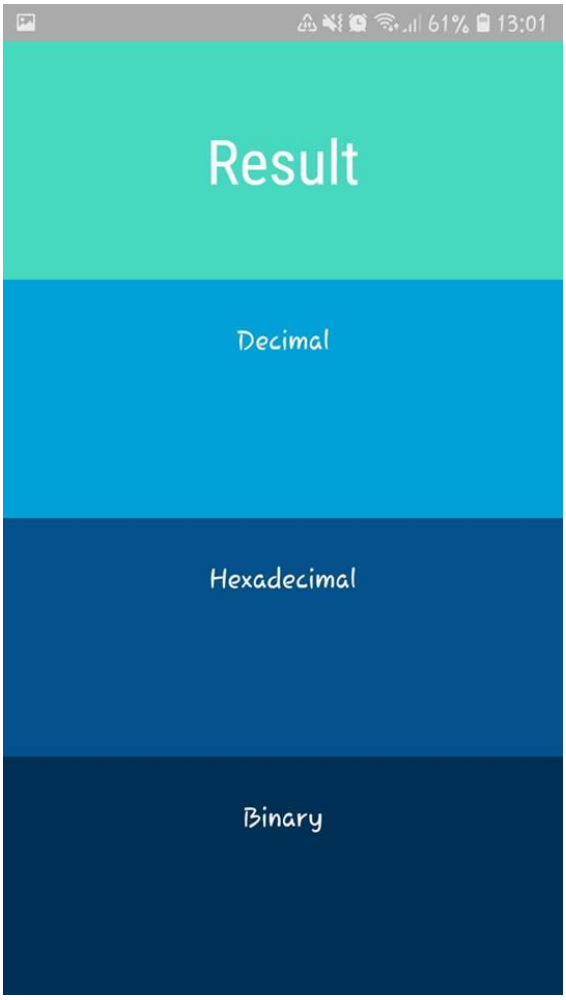
5.2 Μετατροπέας

Κάνοντας κύλιση δεξιά, ο χρήστης μεταφέρεται στον μετατροπέα. Το προεπιλεγμένο αριθμητικό σύστημα είναι το δεκαδικό, ενώ από κάτω του βρίσκεται κατά σειρά, το δεκαεξαδικό, το δυαδικό και ο ASCII κώδικας ο οποίος -προς το παρόν- δεν είναι προς επιλογή για τον χρήστη.

Αφού ο χρήστης επιλέξει το αριθμητικό σύστημα στο οποίο ανήκει ο αριθμός προς μετατροπή, πληκτρολογεί τον αριθμό και αυτόματα θα δει να συμπληρώνονται και τα υπόλοιπα πεδία.

Το αριθμητικό σύστημα που θα επιλέξει καθορίζει και τα πλήκτρα τα οποία θα γίνονται ορατά και προσβάσιμα προς πληκτρολόγηση. Για παράδειγμα, αν επιλέξω το δυαδικό σύστημα (BIN), μόνο το 0 και το 1 παραμένουν άσπρα και προς επιλογή.

5.3 Τελική Οθόνη



Από τον μετατροπέα, ο χρήστης μπορεί να μεταβεί στην αριθμομηχανή με αριστερή κύλιση, ενώ υπάρχει και μια τρίτη οθόνη για την εφαρμογή με μία ακόμα αριστερά κύλιση από την αριθμομηχανή. Πρόκειται ουσιαστικά για μια τελική οθόνη της αριθμομηχανής όπου το αποτέλεσμα που είχαμε σε αυτήν, διακρίνεται σε διαφορετικά αριθμητικά συστήματα.

ΚΕΦΑΛΑΙΟ 6^{ο} :

Προκλήσεις και οι λύσεις τους

6.1 Ανάπτυξη εφαρμογής που προσαρμόζεται σε όλες τις οθόνες

Η αγορά των έξυπνων τηλεφώνων έχει **αναπτυχθεί** αρκετά τα τελευταία χρόνια με αποτέλεσμα να έχει αυξηθεί και **το φάσμα** συσκευών διαφορετικών τεχνικών προδιαγραφών. Αυτό έχει ως αποτέλεσμα η κοινότητα που ασχολείται με τον σχεδιασμό και τη κατασκευή εφαρμογών να είναι αντιμέτωπη με μία μεγάλη πρόκληση. Κι αυτή δεν είναι άλλη από τη δυνατότητα της εφαρμογής να προσαρμόζεται στο μέγεθος της οθόνης κάθε συσκευής παρέχοντας στον χρήστη την ευκολία να αλληλοεπιδρά με της λειτουργίες της.

Ο μεγάλος αριθμός **πλήκτρων** που παρουσιάζεται στη διεπαφή της εφαρμογής, αντιπροσωπεύει τον εξελιγμένο χαρακτήρα της σε σύγκριση με τις αντίστοιχες εφαρμογές.

Άρα για την ομαλή αλληλεπίδραση του χρήστη με τη διεπαφή, ο προσανατολισμός της οθόνης παραμένει κλειδωμένος κάθετα σε όλες τις συσκευές.

6.2 Γρήγορα πρόσβαση στην αριθμομηχανή

Ο αρχικός σχεδιασμός έδινε τη δυνατότητα στον χρήστη, περνώντας από την οθόνη του κεντρικού μενού, να περιηγηθεί στις λειτουργίες της εφαρμογής.

Σαν αποτέλεσμα, ο χρήστης αναγκαζόταν να πραγματοποιήσει επιπλέον ενέργειες σε περίπτωση που επιθυμούσε να μεταβεί από τον μετατροπέα στην αριθμομηχανή και το αντίστροφο. Ο τελικός σχεδιασμός εκμεταλλευόμενος τα Fragments του Android επιτρέπει στον χρήστη την άμεση πρόσβαση στην αριθμομηχανή αλλά και την εύκολη μεταφορά με το σύρσιμο του δακτύλου στον μετατροπέα.

6.3 Χειρισμός Πολικότητας

Μία από τις σημαντικότερες δυνατότητες της εφαρμογής είναι η επιλογή της πολικότητας του αριθμού που **εισάγεται** ανεξάρτητα του επιλεγμένου αριθμητικού συστήματος.

Η αναπαράσταση ενός αρνητικού συστήματος στο δεκαδικό σύστημα είναι μία απλή υπόθεση αλλά στο δυαδικό και δεκαεξαδικό σύστημα μπορεί να γίνει με πολλούς τρόπους

(Συμπλήρωμα ως προς 1, Συμπλήρωμα ως προς 2, Πρόσημο και Μέτρο)

Λαμβάνοντας υπόψη ότι οι υπολογισμοί τόσο στον μετατροπέα αλλά και στην αριθμομηχανή πραγματοποιούνται στη βάση του δεκαδικού συστήματος, οι αριθμοί κατά τη πρώτη προσέγγιση αναπαράστασης τους ως αρνητικοί στο δυαδικό και δεκαεξαδικό σύστημα έκαναν χρήση του Συμπληρώματος ως προς 2. Αυτό είχε σαν αποτέλεσμα κατά την εναλλαγή της πολικότητας, στο παρασκήνιο να πραγματοποιούνται πληθώρα πράξεων και το μήκος του παραγόμενου αριθμού να ξεπερνά τα όρια της διαθέσιμης περιοχής θέασης.

ΚΕΦΑΛΑΙΟ 7^{ο} :

Προοπτικές

Στη παρούσα κατάσταση η εφαρμογή στοχεύει να καλύψει τις ανάγκες για βασικές πράξεις μεταξύ αριθμών που ανήκουν σε αριθμητικά συστήματα, πέραν αυτών που χρησιμοποιούνται στη καθημερινότητα κατά κόρον όπως είναι το δεκαδικό. Επομένως αφορά χρήστες που αποτελούν μέρος κυρίως της επιστημονικής και εκπαιδευτικής κοινότητας. Ας εξετάσουμε λοιπόν μελλοντικές προοπτικές που αφορούν τη λειτουργικότητα της εφαρμογής.

7.1 Λογικές πράξεις

Οι αριθμοί μπορούν να αναπαραστήσουν κομμάτια πληροφορίας άρα η προσθήκη διεπαφής που θα βοηθήσει να πραγματοποιηθούν οι τέσσερις βασικές λογικές πράξεις οι οποίες εκτελούνται μεταξύ δεκαεξαδικών και δυαδικών αριθμών: AND, OR, XOR και NOT θα επεκτείνει τη μέχρι πρότινος λειτουργικότητα.

7.2 Αναπαράσταση αποτελέσματος με χαρακτήρες ASCII

Ο κώδικας ASCII (American Standard Code
for Information Interchange, Αμερικανικός Πρότυπος Κώδικας για Ανταλλαγή Πληροφοριών) είναι ένα κωδικοποιημένο σύνολο χαρακτήρων του λατινικού αλφάβητου και χρησιμοποιείται για αναπαράσταση κειμένου στους υπολογιστές και συσκευές τηλεπικοινωνίας.

Η αναπαράσταση της εισαγόμενης αριθμητικής ακολουθίας στο περιβάλλον του μετατροπέα ή του τελικού αποτελέσματος στο περιβάλλον της αριθμομηχανής με χαρακτήρες ASCII, θα έδινε τη δυνατότητα στο χρήστη να αποκωδικοποιήσει τη πληροφορία μέσω αναγνώσιμου κειμένου.

7.3 Παραδείγματα πράξεων ανάμεσα σε αριθμητικά συστήματα

Το πιο κοινό αριθμητικό σύστημα που χρησιμοποιείται στη καθημερινότητα είναι το δεκαδικό σύστημα. Από τα πρώτα χρόνια της σχολικής σταδιοδρομία διδασκόμαστε πράξεις μεταξύ αριθμών που ανήκουν σε αυτό το σύστημα και έτσι μπορούμε να ερμηνεύσουμε αποτελέσματα κάνοντας επαλήθευση.

Στη περίπτωση της παρούσας εφαρμογής οι πράξεις μεταξύ αριθμών που ανήκουν σε διαφορετικά αριθμητικά συστήματα κρύβει στο παρασκήνιο άλλες διαδοχικές πράξεις και μετατροπές, άγνωστες στους περισσοτέρους χρήστες.

Με γνώμονα τα παραπάνω η ενσωμάτωση παραδειγμάτων αλλά και η παρουσίαση των βηματισμών που πραγματοποιήθηκαν για να προκύψει το αποτέλεσμα μίας πράξης θα εισαγάγουν τον χρήστη στη μεθοδολογία που ακολουθήσαμε.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] - Carl Hammacher, Zvonko Vrasenic and Safwat Zaky, Οργάνωση και Αρχιτεκτονική Ηλεκτρονικών Υπολογιστών, McGraw-Hill 2002,ΕΚΔΟΣΕΙΣΕΠΙΚΕΝΤΡΟ 2006

[2] - Α.Βαφειάδης, Εισαγωγή στην επιστήμη των Υπολογιστών, Θεσσαλονίκη 2003.

[3] - Steve Hollasch, IEEE Standart 754 Floating Point Numbers

[4] - Prof. W. Kahan, IEEE Standart 754 for Binary Floating-Point Arithmetic

[5] - Prof. W. Kahan, Why do we need a floating point arithmetic standart ?

[6] - Frank J. Testa FJT Consulting, IEEE 754 Compliant Floating PointRoutines.

[7]- http://www.math.psu.edu/dna/disasters/patriot.html

[8] - http://babbage.cs.qc.edu/IEEE-754

[9] - http://zeus.it.uom.gr/project/mycomputer/cpu/es_meri2.html

[10] - http://www.lexotypo.com/easyconsole.cfm?page=word_meaning &w_id=400&wlang=gr

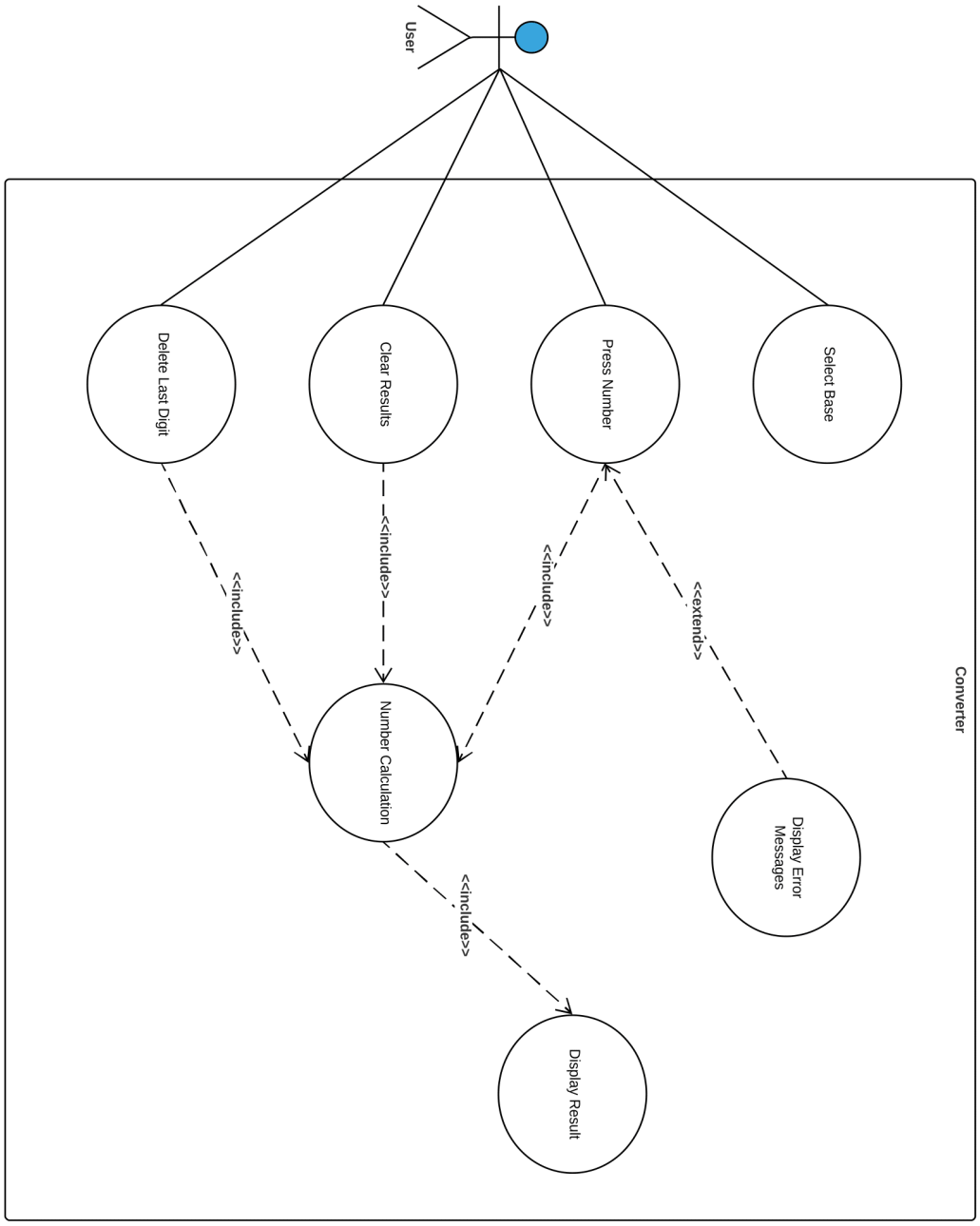
[11] - http://www.mathworks.com/company/pentium/index.shtml

[12] - http://www.cs.uaf.edu/~cs301/notes

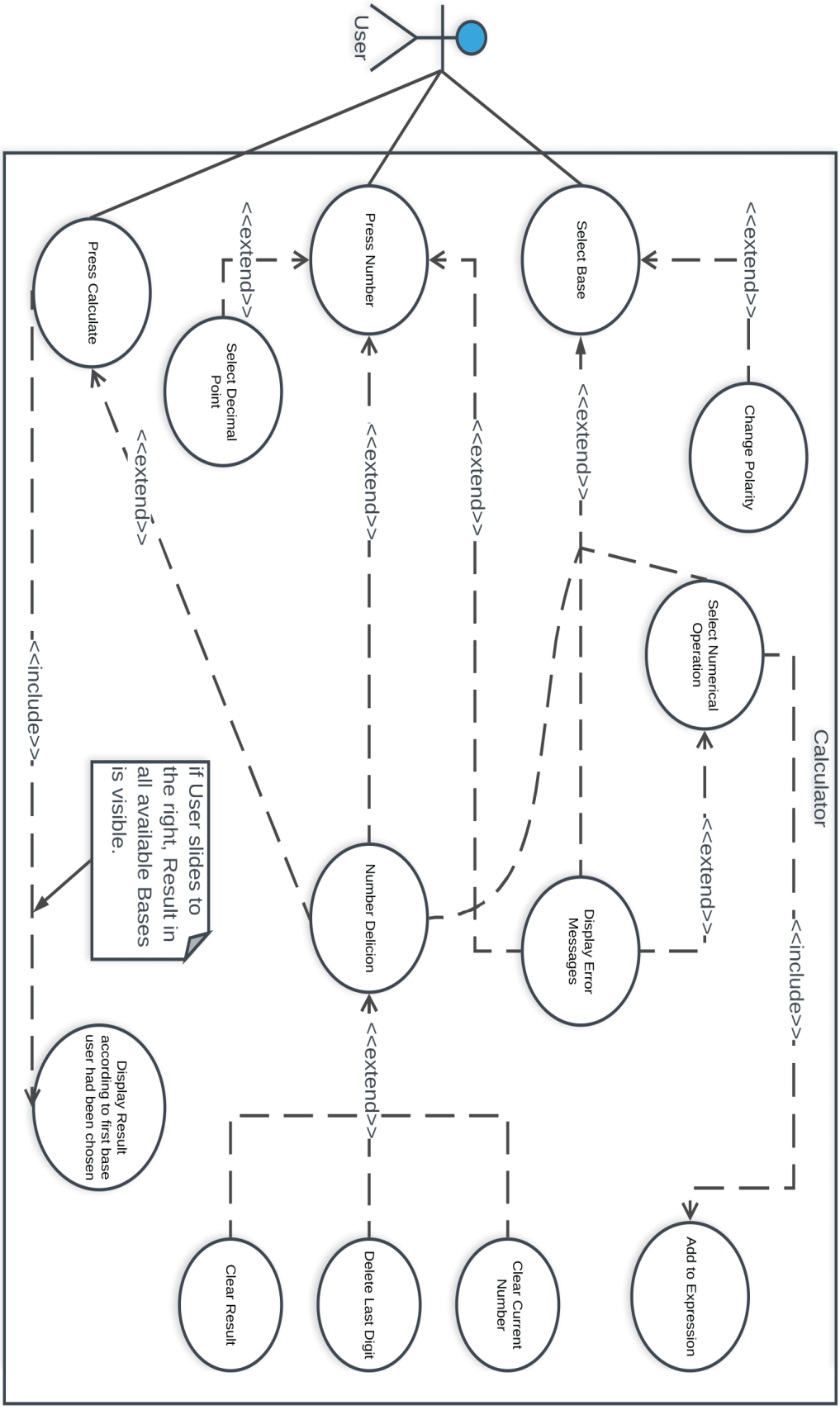
ΠΑΡΑΡΤΗΜΑ Α’ :

Διαγράμματα περιπτώσεων UML

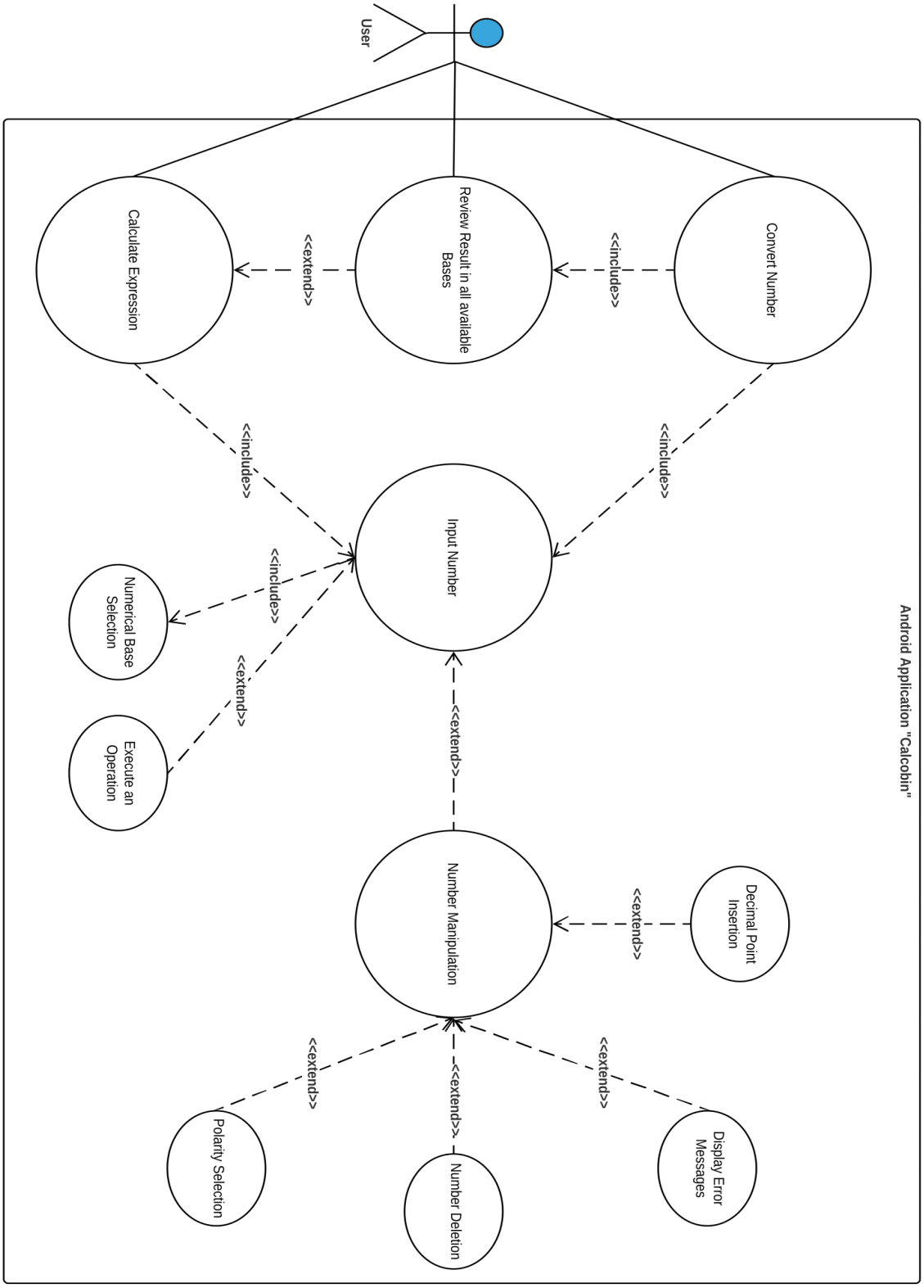
Α.1 Διάγραμμα περιπτώσεων Μετατροπέα



A.2 Διάγραμμα περιπτώσεων Αριθμομηχανής



Α.3 Διάγραμμα περιπτώσεων εφαρμογής



ΠΑΡΑΡΤΗΜΑ Β' :

Ο κώδικας ανάπτυξης της παρούσας εφαρμογής

ANDROIDMANIFEST.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.calcobin">

   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
   <uses-permission android:name="android.permission.INTERNET" />

   <uses-permission android:name="android.permission.WAKE_LOCK" />

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/FullscreenTheme">
      <activity
         android:name=".MainActivity"
         android:label="@string/title_activity_splash"
         android:configChanges="orientation|keyboardHidden|screenSize"
         android:windowSoftInputMode="stateHidden">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
</manifest>
```

BASECONVERTION.JAVA

```
package com.calcobin.helpers;


import java.math.BigDecimal;

import java.math.BigInteger;



public class BaseConvertion {

   public static final char[] EXTENDED = { 0x0080, 0x0081, 0x0082, 0x0083, 0x0084, 0x0085,
0x0086, 0x0087, 0x0088, 0x0089,

                        0x008A, 0x008B, 0x008C, 0x008D, 0x008E, 0x008F, 0x0090, 0x0091,
0x0092, 0x0093,

                        0x0094, 0x0095, 0x0096, 0x0097, 0x0098, 0x0099, 0x009A, 0x009B,
0x009C, 0x009D,

                        0x009E, 0x009F, 0x00A0, 0x00A1, 0x00A2, 0x00A3, 0x00A4, 0x00A5,
0x00A6, 0x00A7,

                        0x00A8, 0x00A9, 0x00AA, 0x00AB, 0x00AC, 0x00AD, 0x00AE,
0x00AF, 0x00B0, 0x00B1,

                        0x00B2, 0x00B3, 0x00B4, 0x00B5, 0x00B6, 0x00B7, 0x00B8, 0x00B9,
0x00BA, 0x00BB,

                        0x00BC, 0x00BD, 0x00BE, 0x00BF, 0x00C0, 0x00C1, 0x00C2, 0x00C3,
0x00C4, 0x00C5,

                        0x00C6, 0x00C7, 0x00C8, 0x00C9, 0x00CA, 0x00CB, 0x00CC, 0x00CD,
0x00CE, 0x00CF,

                        0x00D0, 0x00D1, 0x00D2, 0x00D3, 0x00D4, 0x00D5, 0x00D6,
0x00D7, 0x00D8, 0x00D9,

                        0x00DA, 0x00DB, 0x00DC, 0x00DD, 0x00DE, 0x00DF, 0x00E0,
0x00E1, 0x00E2, 0x00E3,

                        0x00E4, 0x00E5, 0x00E6, 0x00E7, 0x00E8, 0x00E9, 0x00EA, 0x00EB,
0x00EC, 0x00ED,

                        0x00EE, 0x00EF, 0x00F0, 0x00F1, 0x00F2, 0x00F3, 0x00F4, 0x00F5,
0x00F6, 0x00F7,

                        0x00F8, 0x00F9, 0x00FA, 0x00FB, 0x00FC, 0x00FD, 0x00FE, 0x00FF};


   public static final char[] EXTENDED2 = { 0x00C7, 0x00FC, 0x00E9, 0x00E2,
```

```
    0x00E4, 0x00E0, 0x00E5, 0x00E7, 0x00EA, 0x00EB, 0x00E8, 0x00EF,
    0x00EE, 0x00EC, 0x00C4, 0x00C5, 0x00C9, 0x00E6, 0x00C6, 0x00F4,
    0x00F6, 0x00F2, 0x00FB, 0x00F9, 0x00FF, 0x00D6, 0x00DC, 0x00A2,
    0x00A3, 0x00A5, 0x20A7, 0x0192, 0x00E1, 0x00ED, 0x00F3, 0x00FA,
    0x00F1, 0x00D1, 0x00AA, 0x00BA, 0x00BF, 0x2310, 0x00AC, 0x00BD,
    0x00BC, 0x00A1, 0x00AB, 0x00BB, 0x2591, 0x2592, 0x2593, 0x2502,
    0x2524, 0x2561, 0x2562, 0x2556, 0x2555, 0x2563, 0x2551, 0x2557,
    0x255D, 0x255C, 0x255B, 0x2510, 0x2514, 0x2534, 0x252C, 0x251C,
    0x2500, 0x253C, 0x255E, 0x255F, 0x255A, 0x2554, 0x2569, 0x2566,
    0x2560, 0x2550, 0x256C, 0x2567, 0x2568, 0x2564, 0x2565, 0x2559,
    0x2558, 0x2552, 0x2553, 0x256B, 0x256A, 0x2518, 0x250C, 0x2588,
    0x2584, 0x258C, 0x2590, 0x2580, 0x03B1, 0x00DF, 0x0393, 0x03C0,
    0x03A3, 0x03C3, 0x00B5, 0x03C4, 0x03A6, 0x0398, 0x03A9, 0x03B4,
    0x221E, 0x03C6, 0x03B5, 0x2229, 0x2261, 0x00B1, 0x2265, 0x2264,
    0x2320, 0x2321, 0x00F7, 0x2248, 0x00B0, 0x2219, 0x00B7, 0x221A,
    0x207F, 0x00B2, 0x25A0, 0x00A0 };


/* Convertion Without Fraction Part */


public static String FromDecimalToBinary(String input) {
   String binaryResult = null;
   try {

      BigInteger decimalInput = new BigInteger(input);
      binaryResult = decimalInput.toString(2);

   } catch (NumberFormatException numberEx) {
      System.out.print(numberEx);
   }
   return binaryResult;
```

```
}
public static String FromDecimalToHex(String input) {
   String hexResult = null;
   try {

      BigInteger decimalInput = new BigInteger(input);
      hexResult = decimalInput.toString(16);

   } catch (NumberFormatException numberEx) {
      System.out.print(numberEx);
   }
   return hexResult;
}

public static StringBuilder FromDecimalToAscii(String input) {
   StringBuilder asciiResult = null;
   try {




      BigInteger decimalInput = new BigInteger(input);
      String hexResult = decimalInput.toString(16);

      asciiResult = new StringBuilder("");

      for (int i = 0; i < hexResult.length() - 1; i += 2) {
         String output = hexResult.substring(i, (i + 2));
         int decimal = Integer.parseInt(output, 16);
         if (decimal >= 0x80 && decimal <= 0xFF) {
            int temp = decimal - 128;
            char tempschar = EXTENDED2[temp];
```

```
            asciiResult.append(tempschar);
        }else{
            asciiResult.append((char) decimal);
        }
      }

   } catch (NumberFormatException numberEx) {
      System.out.print(numberEx);
   }
   return asciiResult;
}

public static String FromHexToDecimal(String input) {
   String decimalResult = null;
   try {

      BigInteger hexInput = new BigInteger(input, 16);
      decimalResult = hexInput.toString();

   } catch (NumberFormatException numberEx) {
      System.out.print(numberEx);
   }
   return decimalResult;
}

public static String FromHexToBinary(String input) {
   String binaryResult = null;
   try {

      BigInteger hexInput = new BigInteger(input, 16);
      binaryResult = hexInput.toString(2);
```

```
    } catch (NumberFormatException numberEx) {
        System.out.print(numberEx);
    }
    return binaryResult;
}

public static StringBuilder FromHexToAscii(String input) {
    StringBuilder asciiResult = null;
    try {

        asciiResult = new StringBuilder("");

        for (int i = 0; i < input.length()-1; i += 2){
            String output = input.substring(i, (i + 2));
            int decimal = Integer.parseInt(output, 16);
            if (decimal >= 0x80 && decimal <= 0xFF) {
                int temp = decimal - 128;
                char tempschar = EXTENDED2[temp];
                asciiResult.append(tempschar);
            }else{
                asciiResult.append((char) decimal);
            }
            //asciiResult.append((char) Integer.parseInt(output, 16));
        }

    } catch (NumberFormatException numberEx) {
        System.out.print(numberEx);
    }
    return asciiResult;
}

public static String FromBinaryToDecimal(String input) {
```

```
    String decimalResult = null;
    try {

        BigInteger binaryInput = new BigInteger(input, 2);
        decimalResult = binaryInput.toString();

    } catch (NumberFormatException numberEx) {
        System.out.print(numberEx);
    }
    return decimalResult;
}

public  static String FromBinaryToHex(String input) {
    String hexResult = null;
    try {

        BigInteger binaryInput = new BigInteger(input, 2);
        hexResult = binaryInput.toString(16);

    } catch (NumberFormatException numberEx) {
        System.out.print(numberEx);
    }
    return hexResult;
}

public static StringBuilder FromBinaryToAscii(String input){
    StringBuilder asciiResult = null;
    try{

        BigInteger binaryInput = new BigInteger(input, 2);
        String hexResult = binaryInput.toString(16);
        asciiResult = new StringBuilder("");
```

```
        for (int i = 0; i < hexResult.length()-1; i += 2)
        {
            String output = hexResult.substring(i, (i + 2));
            int decimal = Integer.parseInt(output, 16);
            if (decimal >= 0x80 && decimal <= 0xFF) {
                int temp = decimal - 128;
                char tempschar = EXTENDED2[temp];
                asciiResult.append(tempschar);
            }else{
                asciiResult.append((char) decimal);
            }
            //asciiResult.append((char) Integer.parseInt(output, 16));
        }

    }catch (NumberFormatException numberEx) {
        System.out.print(numberEx);
    }
    return asciiResult;
}
```

```
/* Convertion With Fraction Part */

public static StringBuilder FromDecimalToBinaryWithFraction(String input) {
    StringBuilder str = null;
    try {
        Double decimal = Double.parseDouble(input);
        int n = 32;  // constant?
```

```
        BigDecimal bd = new BigDecimal(decimal);
        BigDecimal mult = new BigDecimal(2).pow(n);
        bd = bd.multiply(mult);
        BigInteger bi = bd.toBigInteger();
        str = new StringBuilder(bi.toString(2));
        while (str.length() < n + 1) {  // +1 for leading zero
           str.insert(0, "0");
        }
        str.insert(str.length() - n, ".");
        for(int i=str.length(); i>0; i--){
           if(str.charAt(str.length() - 1) == '0'){
              str.setLength(str.length()-1);
           }
           if(str.charAt(str.length() - 1) == '.'){
              str.setLength(str.length()-1);
           }
        }
     } catch (NumberFormatException numberEx) {
        System.out.print(numberEx);
     }
     return str;
  }

  public static  StringBuilder FromDecimalToHexWithFraction(String input) {
     StringBuilder hexOut = null;
     try {
        hexOut = new StringBuilder();
        double doubleOfDecInp = Double.parseDouble(input);

        if (doubleOfDecInp < 0) {

           hexOut = hexOut.append("-");
```

```
            doubleOfDecInp = -doubleOfDecInp;
        }

        BigInteger beforedot = new BigDecimal(doubleOfDecInp).toBigInteger();
        hexOut.append(beforedot.toString(16));

        BigDecimal bfd = new BigDecimal(beforedot);
        doubleOfDecInp = doubleOfDecInp - bfd.doubleValue();

        if (doubleOfDecInp == 0) {
            hexOut.toString();
        }
        hexOut.append(".");

        for (int i = 0; i < 16; ++i) {
            doubleOfDecInp = doubleOfDecInp * 16;
            int digit = (int) doubleOfDecInp;

            hexOut.append(Integer.toHexString(digit));

            doubleOfDecInp = doubleOfDecInp - digit;

            if (doubleOfDecInp == 0)
                break;
        }

    } catch (NumberFormatException numberEx) {
        System.out.print(numberEx);
    }
    return hexOut;
}
```

```
    public static String FromHexToDecimalWithFraction(String input) {
        String fromhextodec = null;
        try {
            fromhextodec = Double.valueOf(input).toString();
        } catch (NumberFormatException numberEx) {
            System.out.print(numberEx);
        }
        return fromhextodec;
    }

    public static String FromBinaryToDecimalWithFraction(String input) {
        String resultDec = null;
        try {
            String[] splits = input.split("\\.");
            String beforeDot = splits[0];
            String afterDot = splits[1];

            Integer asIntBeforeForDecimal = Integer.parseInt(beforeDot, 2);
            Integer placesAfterPoint = afterDot.length() - afterDot.indexOf(".") - 1;
            Long numerator = Long.parseLong(afterDot.replace(".", ""), 2);
            Double value = ((double) numerator) / (1L << placesAfterPoint);

            Double result = asIntBeforeForDecimal + value;
            resultDec = String.valueOf(result);
        } catch (NumberFormatException numberEx) {
            System.out.print(numberEx);
        }
        return resultDec;
    }

}
```

BASEFORMATER.JAVA

```
package com.calcobin.helpers;

import android.text.Spannable;
import android.text.SpannableStringBuilder;
import android.text.style.RelativeSizeSpan;
import android.text.style.SubscriptSpan;

public class BaseFormater {

   public static boolean isDecimalFast(String s)
   {
      boolean valid = true;
      char[] a = s.toCharArray();
      for (char c: a)
      {
         valid = ((c >= '0') && (c <= '9')) ||
               ((c >= '.') && (c <= '.'));
         if (!valid)
         {
            break;
         }
      }
      return valid;
   }

   public static boolean isHexFast(String s)
   {
      boolean valid = true;
      char[] a = s.toCharArray();
      for (char c: a)
      {
         valid = ((c >= 'A') && (c <= 'F')) ||
               ((c >= '0') && (c <= '9')) ||
               ((c >= '.') && (c <= '.'));
         if (!valid)
         {
            break;
         }
      }
      return valid;
   }

   public static boolean isBinFast(String s)
   {
      boolean valid = true;
      char[] a = s.toCharArray();
```

```
        for (char c: a)
        {
            valid = ((c >= '0') && (c <= '1')) ||
                    ((c >= '.') && (c <= '.'));
            if (!valid)
            {
                break;
            }
        }
        return valid;
    }

    public static String clearBase(String input, String currentBase) {
        String clearedInput = input.substring(0, input.length() - currentBase.length());//clear
Base
        return clearedInput;
    }

    public static SpannableStringBuilder format(String input, String base){
        String formingInput = input + base;
        SpannableStringBuilder formattedInput = new SpannableStringBuilder(formingInput);
        formattedInput.setSpan(new SubscriptSpan(), formattedInput.length()-base.length(),
formattedInput.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
        formattedInput.setSpan(new RelativeSizeSpan(0.5f), formattedInput.length()-
base.length(), formattedInput.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
        return formattedInput;
    }

}
```

ERRORHANDLER.JAVA

```
package com.calcobin.helpers;

import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.calcobin.R;

public class ErrorHandler {

   public static void echoError(ErrorType mCurrentError, Activity currentActivity){
      String errorMessage = null;
      switch (mCurrentError){
         case LENGTH:
            errorMessage = "Too big number";
            break;
         case BASE:
            errorMessage = "Please select a valid base";
            break;
         case FULLSTACK:
            errorMessage = "Stack is full. Please hit enter to proceed";
            break;
         case DIVIDEWITHZERO:
            errorMessage = "You cannot divide with zero";
            break;
         case DEC_LENGTH:
            errorMessage = "Too big Decimal input";
            break;
         case HEX_LENGTH:
            errorMessage = "Too big Hexadecimal input";
            break;
         case BIN_LENGTH:
            errorMessage = "Too big Binary input";
            break;
      }
      Toast.makeText(currentActivity, errorMessage,
            Toast.LENGTH_LONG).show();
   }

   public static void rotateDevice(Context context, Activity currentActivity) {
      LayoutInflater inflater = currentActivity.getLayoutInflater();
            View view = inflater.inflate(R.layout.rotate_toast,
                  (ViewGroup) currentActivity.findViewById(R.id.overlay));
```

```
            Toast toast = new Toast(context);
            toast.setView(view);
            toast.show();
    }

    public enum ErrorType{
        NULL, LENGTH, BASE, FULLSTACK, DIVIDEWITHZERO, DEC_LENGTH, HEX_LENGTH,
BIN_LENGTH, ROTATE
    }

}
```

PLUSMINUS.JAVA

```
package com.calcobin.helpers;


import android.text.SpannableStringBuilder;

import com.calcobin.Calculator;

import static java.lang.Math.pow;

public class PlusMinus {

   private static SpannableStringBuilder fixedInput;

   public static void DecimalPolarity(Calculator.NumberPolarity polarity, String input) {
      String clearBase, clearInput, fixPolarity="";
      fixedInput = new SpannableStringBuilder("");
      if (polarity.equals(Calculator.NumberPolarity.POSITIVE)) {
         clearBase = input.substring(0, input.length() - Calculator.currentBase.length());
         //clearInput = clearBase.replaceAll("[()]", "");
         //fixPolarity = clearInput.replaceAll("[-]","");
         fixPolarity = clearBase.replaceAll("[-]","");
         Calculator.convertedInput = Calculator.convertedInput.replaceAll("[-]","");
         fixedInput.clear();
         fixedInput = BaseFormater.format(fixPolarity,Calculator.currentBase);
         Calculator.formatedInput = fixedInput;
         Calculator.editTextInput.setText(fixedInput);
         //fixedInput.clear();
      } else if (polarity.equals(Calculator.NumberPolarity.NEGATIVE)) {
         clearBase = input.substring(0, input.length() - Calculator.currentBase.length());
         //clearInput = clearBase.replaceAll("[()]", "");
         //fixPolarity = "-" + clearInput;
         fixPolarity = "-" + clearBase;
         Calculator.convertedInput = "-" + Calculator.convertedInput;
         fixedInput.clear();
         fixedInput = BaseFormater.format(fixPolarity,Calculator.currentBase);
         Calculator.formatedInput = fixedInput;
         Calculator.editTextInput.setText(fixedInput);
         //fixedInput.clear();
         String here = "here";
      } else {
      }
   }

   public static void BinaryPolarity(Calculator.NumberPolarity polarity, String input){
      String clearBase, clearInput, fixPolarity;
      fixedInput = new SpannableStringBuilder("");
```

```
        if (polarity.equals(Calculator.NumberPolarity.POSITIVE)) {
            Calculator.convertedInput = Calculator.convertedInput.replaceAll("[-]","");
            if (Calculator.convertedInput.contains(".")) {
                fixPolarity =
String.valueOf(BaseConvertion.FromDecimalToBinaryWithFraction(Calculator.convertedInpu
t));
            } else {
                fixPolarity = BaseConvertion.FromDecimalToBinary(Calculator.convertedInput);
            }
            fixedInput.clear();
            fixedInput = BaseFormater.format(fixPolarity,Calculator.currentBase);
            Calculator.formatedInput = fixedInput;
            Calculator.editTextInput.setText(fixedInput);
        } else if (polarity.equals(Calculator.NumberPolarity.NEGATIVE)){
            clearBase = input.substring(0, input.length() - Calculator.currentBase.length());
            //clearInput = clearBase.replaceAll("[()]", "");
            if (Calculator.convertedInput.contains(".")) {
                fixPolarity = "-" + clearBase;
                Calculator.convertedInput = "-" + Calculator.convertedInput;
                fixedInput.clear();
                fixedInput = BaseFormater.format(fixPolarity,Calculator.currentBase);
                Calculator.formatedInput = fixedInput;
                Calculator.editTextInput.setText(fixedInput);
            } else {
                fixPolarity = "-" + clearBase;
                Calculator.convertedInput = "-" + Calculator.convertedInput;
                fixedInput.clear();
                fixedInput = BaseFormater.format(fixPolarity,Calculator.currentBase);
                Calculator.formatedInput = fixedInput;
                Calculator.editTextInput.setText(fixedInput);
            }
        } else {
        }
    }

    public static void HexPolarity(Calculator.NumberPolarity polarity, String input) {
        String clearBase, clearInput, fixPolarity, takeBinaryExpression;
        fixedInput = new SpannableStringBuilder("");
        if (polarity.equals(Calculator.NumberPolarity.POSITIVE)) {
            Calculator.convertedInput = Calculator.convertedInput.replaceAll("[-]","");
            if (Calculator.convertedInput.contains(".")) {
                fixPolarity =
BaseConvertion.FromDecimalToHexWithFraction(Calculator.convertedInput).toString().toUp
perCase();
            } else {
                fixPolarity =
BaseConvertion.FromDecimalToHex(Calculator.convertedInput).toUpperCase();
            }
            fixedInput.clear();
```

```
            fixedInput = BaseFormater.format(fixPolarity,Calculator.currentBase);
            Calculator.formatedInput = fixedInput;
            Calculator.editTextInput.setText(fixedInput);
        } else if (polarity.equals(Calculator.NumberPolarity.NEGATIVE)) {
            clearBase = input.substring(0, input.length() - Calculator.currentBase.length());
            //clearInput = clearBase.replaceAll("[()]", "");
            if (Calculator.convertedInput.contains(".")) {
                String hexformated = "0x"+ clearBase + "p0";
                String decfromhex = Double.valueOf(hexformated).toString();
                StringBuilder binaryResult =
BaseConvertion.FromDecimalToBinaryWithFraction(decfromhex);
                takeBinaryExpression = binaryResult.toString();
                //takeBinaryExpression = negativeBinaryWithFraction(binaryResult.toString());
                String decResult =
BaseConvertion.FromBinaryToDecimalWithFraction(takeBinaryExpression);
                StringBuilder hexResult =
BaseConvertion.FromDecimalToHexWithFraction(decResult);
                fixPolarity = "-" + hexResult.toString().toUpperCase();
                //fixPolarity = hexResult.toString().toUpperCase();
                Calculator.convertedInput = "-" + Calculator.convertedInput;
                fixedInput.clear();
                fixedInput = BaseFormater.format(fixPolarity,Calculator.currentBase);
                Calculator.formatedInput = fixedInput;
                Calculator.editTextInput.setText(fixedInput);
            } else {
                String binaryResult = BaseConvertion.FromHexToBinary(clearBase);
                takeBinaryExpression = binaryResult;
                fixPolarity = "-" +
BaseConvertion.FromBinaryToHex(takeBinaryExpression).toUpperCase();
                //takeBinaryExpression = negativeBinary(binaryResult);
                //fixPolarity =
BaseConvertion.FromBinaryToHex(takeBinaryExpression).toUpperCase();
                Calculator.convertedInput = "-" + Calculator.convertedInput;
                fixedInput.clear();
                fixedInput = BaseFormater.format(fixPolarity,Calculator.currentBase);
                Calculator.formatedInput = fixedInput;
                Calculator.editTextInput.setText(fixedInput);
            }
        }
    }

    public static String negativeBinary(String clearInput) {
        String twosComplement="";
        int n = clearInput.length();
        Double subtracter = pow(2,n);
        Double subtraction = subtracter - Double.parseDouble(Calculator.convertedInput);
        String subtractionString = String.valueOf(subtraction);
        if (subtractionString.contains(".")) {
            if (Double.parseDouble(subtractionString) != 0) {
```

```
            String[] splits = subtractionString.split("\\.");
            String beforeDot = splits[0];
            String afterDot = splits[1];
            if (Double.parseDouble(afterDot) == 0) {
               if (Double.parseDouble(beforeDot) != 0) {
                  twosComplement = BaseConvertion.FromDecimalToBinary(beforeDot);
                  for (int i=twosComplement.length(); i<n; i++) {
                     twosComplement = "0" + twosComplement;
                  }
               }
               for (int i=twosComplement.length(); i<8; i++) {
                  twosComplement = "1" + twosComplement;
               }
            }
         } else {
         }
      }
      return twosComplement;
   }

   public static String negativeBinaryWithFraction(String clearInput) {
      String temp1 = clearInput.replaceAll("0","x");
      String temp2 = temp1.replaceAll("1","0");
      String flipped = temp2.replaceAll("x","1");

      String [] splits = flipped.split("\\.");
      String beforeDot = splits[0];
      String afterDot = splits[1];

      String addOne = "1";
      int number0 = Integer.parseInt(afterDot, 2);
      int number1 = Integer.parseInt(addOne, 2);

      String fraction = BaseConvertion.FromDecimalToBinary(Integer.toString(number0 +
number1));
      if (fraction.length() != afterDot.length()) {
         for (int i=fraction.length(); i<afterDot.length(); i++) {
            fraction = "0" + fraction;
         }
      }

      int beforeDotLenght = beforeDot.length();

      if (beforeDotLenght <= 7) {
         for (int i=beforeDot.length(); i<8; i++) {
            beforeDot = "1" + beforeDot;
         }
      } else if (beforeDotLenght <= 15) {
         for (int i=beforeDot.length(); i<16; i++) {
```

```
            beforeDot = "1" + beforeDot;
          }
        } else if (beforeDotLenght <= 31) {
          for (int i=beforeDot.length(); i<32; i++) {
            beforeDot = "1" + beforeDot;
          }
        } else if (beforeDotLenght <= 39) {
          for (int i=beforeDot.length(); i<40; i++) {
            beforeDot = "1" + beforeDot;
          }
        } else if (beforeDotLenght <= 47) {
          for (int i=beforeDot.length(); i<48; i++) {
            beforeDot = "1" + beforeDot;
          }
        } else if (beforeDotLenght <= 55) {
          for (int i=beforeDot.length(); i<56; i++) {
            beforeDot = "1" + beforeDot;
          }
        } else if (beforeDotLenght <= 63) {
          for (int i=beforeDot.length(); i<64; i++) {
            beforeDot = "1" + beforeDot;
          }
        }

        String negativeBinary = beforeDot + "." + fraction;

        return negativeBinary;

    }

}//end
```

CALCULATOR.JAVA

```
package com.calcobin;

//TODO 1) Pass spanable inputs to numberexpression stack CHECK, 2) Change "=" functionality CHECK, 3) Lock stack when it is full CHECK, 4) Display result in editTextInput with the base of the first number CHECK

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.text.Editable;
import android.text.SpannableStringBuilder;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;

import com.calcobin.helpers.BaseConvertion;
import com.calcobin.helpers.PlusMinus;
import com.calcobin.helpers.BaseFormater;
import com.calcobin.helpers.ErrorHandler;

public class Calculator extends Fragment implements View.OnClickListener {

   public static EditText editTextExpession, editTextInput;

   private Button buttonDec,buttonHex, buttonBin, buttonPoint, buttonBackspace, buttonClear, buttonCE, buttonEqual, button0,
         button1, button2, button3, button4, button5, button6, button7, button8, button9,
         buttonA, buttonB, buttonC, buttonD, buttonE, buttonF, buttonPlus, buttonMinus,
         buttonMultiply, buttonDivide, buttonPlusMinus;

   private SpannableStringBuilder numericExpression, expressionCarrier;

   public static SpannableStringBuilder formatedInput;

   public static String currentBase, supremBase, convertedInput = "0", currentSign = "+";

   private Double result=0d;

   private CalculatorState mCurrentState;

   private NumberPolarity currentPolarity;

   private ErrorHandler.ErrorType mCurrentError;
```

```
    private boolean hasBase, fullNumStack;

    private int maxLengthforDec, maxLengthforHex, maxLengthforBin, numInStack;



    final TextWatcher inputTextWatcher = new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {

        }

        @Override
        public void afterTextChanged(Editable s) {

            String input = s.toString();

            if(input.endsWith(".")){
                buttonPoint.setOnClickListener(null);
            }else if(input.contains(".")){
                buttonPoint.setOnClickListener(null);
            }else{
                buttonPoint.setOnClickListener(Calculator.this);
            }
        }
    };

    @Override
    public void onPause() {
        super.onPause();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View v = inflater.inflate(R.layout.activity_calculator, container, false);

        editTextInput = (EditText) v.findViewById(R.id.editTextInput);
        editTextExpession = (EditText) v.findViewById(R.id.editTextExpession);

        buttonDec = (Button) v.findViewById(R.id.decimalNum);
        buttonHex = (Button) v.findViewById(R.id.hexNum);
        buttonBin = (Button) v.findViewById(R.id.binaryNum);
```

```
buttonPoint = (Button) v.findViewById(R.id.dot);
buttonCE = (Button) v.findViewById(R.id.ce);
buttonClear = (Button) v.findViewById(R.id.clear);
buttonBackspace = (Button) v.findViewById(R.id.backspace);
buttonEqual = (Button) v.findViewById(R.id.equal);

button0 = (Button) v.findViewById(R.id.zero);
button1 = (Button) v.findViewById(R.id.one);
button2 = (Button) v.findViewById(R.id.two);
button3 = (Button) v.findViewById(R.id.three);
button4 = (Button) v.findViewById(R.id.four);
button5 = (Button) v.findViewById(R.id.five);
button6 = (Button) v.findViewById(R.id.six);
button7 = (Button) v.findViewById(R.id.seven);
button8 = (Button) v.findViewById(R.id.eigth);
button9 = (Button) v.findViewById(R.id.nine);

buttonA = (Button) v.findViewById(R.id.alpha);
buttonB = (Button) v.findViewById(R.id.beta);
buttonC = (Button) v.findViewById(R.id.gamma);
buttonD = (Button) v.findViewById(R.id.delta);
buttonE = (Button) v.findViewById(R.id.epsilon);
buttonF = (Button) v.findViewById(R.id.fi);

buttonPlus = (Button) v.findViewById(R.id.plus);
buttonMinus = (Button) v.findViewById(R.id.minus);
buttonMultiply = (Button) v.findViewById(R.id.multiply);
buttonDivide = (Button) v.findViewById(R.id.divide);
buttonPlusMinus = (Button) v.findViewById(R.id.plus_minus);

buttonDec.setOnClickListener(this);
buttonHex.setOnClickListener(this);
buttonBin.setOnClickListener(this);

buttonPoint.setOnClickListener(this);
buttonCE.setOnClickListener(this);
buttonClear.setOnClickListener(this);
buttonBackspace.setOnClickListener(this);
buttonEqual.setOnClickListener(this);

button0.setOnClickListener(this);
button1.setOnClickListener(this);
button2.setOnClickListener(this);
button3.setOnClickListener(this);
button4.setOnClickListener(this);
button5.setOnClickListener(this);
button6.setOnClickListener(this);
button7.setOnClickListener(this);
```

```
    button8.setOnClickListener(this);
    button9.setOnClickListener(this);

    buttonA.setOnClickListener(this);
    buttonB.setOnClickListener(this);
    buttonC.setOnClickListener(this);
    buttonD.setOnClickListener(this);
    buttonE.setOnClickListener(this);
    buttonF.setOnClickListener(this);

    buttonPlus.setOnClickListener(this);
    buttonMinus.setOnClickListener(this);
    buttonMultiply.setOnClickListener(this);
    buttonDivide.setOnClickListener(this);
    buttonPlusMinus.setOnClickListener(this);

    initialSetup();

    editTextInput.setSelection(editTextInput.getText().length());
    editTextInput.setFocusable(false);
    editTextInput.setClickable(false);
    editTextInput.setTextIsSelectable(false);
    editTextInput.setFocusableInTouchMode(false);
    editTextInput.setKeyListener(null);
    editTextInput.setEnabled(true);

    editTextInput.addTextChangedListener(inputTextWatcher);

    editTextExpession.setTextColor(getResources().getColor(R.color.disable_gray));
    editTextExpession.setText(null);
    editTextExpession.setFocusable(false);
    editTextExpession.setClickable(false);
    editTextExpession.setTextIsSelectable(false);
    editTextExpession.setFocusableInTouchMode(false);
    editTextExpession.setKeyListener(null);
    editTextExpession.setEnabled(true);

    return v;
  }//telos onCreateView <--


  @Override
  public void onClick(View v) {
    switch (v.getId()) {
      case R.id.backspace:
        onDelete();
        break;
      case R.id.clear:
        // setState(CalculatorState.READY);
```

```
            onClear();
            break;
        case R.id.ce:
            setState(CalculatorState.READY);
            onNumberReset();
            break;
        case R.id.dot:
            setState(CalculatorState.INPUT);
            insert(((Button) v).getText().toString());
            break;
        case R.id.plus_minus:
            if (getPolarity() == NumberPolarity.POSITIVE) {
                setPolarity(NumberPolarity.NEGATIVE);
            } else {
                setPolarity(NumberPolarity.POSITIVE);
            }
            plus_minus(currentBase);
            break;
        case R.id.plus:
            if (getState() == CalculatorState.EVALUATE) {
                pushToResult(currentSign);
            }
            if (getState() != CalculatorState.ERROR) {
                plus(currentSign);
                if (!fullNumStack) {
                    setState(CalculatorState.READY);
                }
            }
            setPolarity(NumberPolarity.POSITIVE);
            break;
        case R.id.minus:
            if (getState() == CalculatorState.EVALUATE) {
                pushToResult(currentSign);
            }
            if (getState() != CalculatorState.ERROR) {
                minus(currentSign);
                if (!fullNumStack) {
                    setState(CalculatorState.READY);
                }
            }
            setPolarity(NumberPolarity.POSITIVE);
            break;
        case R.id.multiply:
            if (getState() == CalculatorState.EVALUATE) {
                pushToResult(currentSign);
            }
            if (getState() != CalculatorState.ERROR) {
                multiply(currentSign);
                if (!fullNumStack) {
```

```
            setState(CalculatorState.READY);
        }
    }
    setPolarity(NumberPolarity.POSITIVE);
    break;
case R.id.divide:
    if (getState() == CalculatorState.EVALUATE) {
        pushToResult(currentSign);
    }
    if (getState() != CalculatorState.ERROR) {
        divide(currentSign);
        if (!fullNumStack) {
            setState(CalculatorState.READY);
        }
    }
    setPolarity(NumberPolarity.POSITIVE);
    break;
case R.id.decimalNum:
    setState(CalculatorState.EVALUATE);
    toDecimal(editTextInput.getText().toString());
    break;
case R.id.hexNum:
    setState(CalculatorState.EVALUATE);
    toHex();
    break;
case R.id.binaryNum:
    setState(CalculatorState.EVALUATE);
    toBin();
    break;
case R.id.equal:
    if (getState() == CalculatorState.EVALUATE) {
        pushToResult(currentSign);
    }
    result();
    break;
default:
    if (expressionCarrier.length() > 0) {
        setPolarity(NumberPolarity.POSITIVE);//check
        fullNumStack = false;
        editTextInput.setText("0");
        hasBase = false;
        numericExpression = new SpannableStringBuilder("");
        pushToNumericExpression(expressionCarrier, InputType.NUMBER);
        expressionCarrier.clear();
        formatedInput.clear();
    }
    if (getState() == CalculatorState.ERROR) {
        onClear();
    } else if ((getState() == CalculatorState.READY) && currentSign == null) {
```

```
            onClear();
        }
        insert(((Button) v).getText().toString());
        setState(CalculatorState.INPUT);
        break;
    }
}


@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

protected CalculatorState getState() {
    return mCurrentState;
}

protected void setState(CalculatorState state) {
    if (mCurrentState != state) {
        mCurrentState = state;
        if (mCurrentState.equals(CalculatorState.READY)) {
            buttonPoint.setOnClickListener(this);
            button0.setOnClickListener(this);
            button1.setOnClickListener(this);
            button2.setOnClickListener(this);
            button3.setOnClickListener(this);
            button4.setOnClickListener(this);
            button5.setOnClickListener(this);
            button6.setOnClickListener(this);
            button7.setOnClickListener(this);
            button8.setOnClickListener(this);
            button9.setOnClickListener(this);
            buttonA.setOnClickListener(this);
            buttonB.setOnClickListener(this);
            buttonC.setOnClickListener(this);
            buttonD.setOnClickListener(this);
            buttonE.setOnClickListener(this);
            buttonF.setOnClickListener(this);
```

```
        buttonPlusMinus.setOnClickListener(null);
        buttonPlus.setOnClickListener(this);
        buttonMinus.setOnClickListener(this);
        buttonMultiply.setOnClickListener(this);
        buttonDivide.setOnClickListener(this);
        buttonEqual.setOnClickListener(this);
    } else if(mCurrentState.equals(CalculatorState.INPUT)) {
        hasBase = false;
        buttonPoint.setOnClickListener(this);
        button0.setOnClickListener(this);
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
        button3.setOnClickListener(this);
        button4.setOnClickListener(this);
        button5.setOnClickListener(this);
        button6.setOnClickListener(this);
        button7.setOnClickListener(this);
        button8.setOnClickListener(this);
        button9.setOnClickListener(this);
        buttonA.setOnClickListener(this);
        buttonB.setOnClickListener(this);
        buttonC.setOnClickListener(this);
        buttonD.setOnClickListener(this);
        buttonE.setOnClickListener(this);
        buttonF.setOnClickListener(this);
        buttonDec.setOnClickListener(this);
        buttonHex.setOnClickListener(this);
        buttonBin.setOnClickListener(this);
        buttonPlusMinus.setOnClickListener(null);
        buttonPlus.setOnClickListener(null);
        buttonMinus.setOnClickListener(null);
        buttonMultiply.setOnClickListener(null);
        buttonDivide.setOnClickListener(null);
        buttonBackspace.setOnClickListener(this);
        buttonEqual.setOnClickListener(null);
    } else if (mCurrentState.equals(CalculatorState.EVALUATE)) {
        buttonPoint.setOnClickListener(null);
        button0.setOnClickListener(null);
        button1.setOnClickListener(null);
        button2.setOnClickListener(null);
        button3.setOnClickListener(null);
        button4.setOnClickListener(null);
        button5.setOnClickListener(null);
        button6.setOnClickListener(null);
        button7.setOnClickListener(null);
        button8.setOnClickListener(null);
        button9.setOnClickListener(null);
        buttonA.setOnClickListener(null);
        buttonB.setOnClickListener(null);
```

```
        buttonC.setOnClickListener(null);
        buttonD.setOnClickListener(null);
        buttonE.setOnClickListener(null);
        buttonF.setOnClickListener(null);
        buttonPlusMinus.setOnClickListener(this);
        buttonPlus.setOnClickListener(this);
        buttonMinus.setOnClickListener(this);
        buttonMultiply.setOnClickListener(this);
        buttonDivide.setOnClickListener(this);
        buttonEqual.setOnClickListener(this);
    } else if (mCurrentState.equals(CalculatorState.RESULT)) {
        buttonPoint.setOnClickListener(null);
        button0.setOnClickListener(null);
        button1.setOnClickListener(null);
        button2.setOnClickListener(null);
        button3.setOnClickListener(null);
        button4.setOnClickListener(null);
        button5.setOnClickListener(null);
        button6.setOnClickListener(null);
        button7.setOnClickListener(null);
        button8.setOnClickListener(null);
        button9.setOnClickListener(null);
        buttonA.setOnClickListener(null);
        buttonB.setOnClickListener(null);
        buttonC.setOnClickListener(null);
        buttonD.setOnClickListener(null);
        buttonE.setOnClickListener(null);
        buttonF.setOnClickListener(null);
        buttonDec.setOnClickListener(null);
        buttonHex.setOnClickListener(null);
        buttonBin.setOnClickListener(null);
        buttonPlusMinus.setOnClickListener(this);
        buttonPlus.setOnClickListener(this);
        buttonMinus.setOnClickListener(this);
        buttonMultiply.setOnClickListener(this);
        buttonDivide.setOnClickListener(this);
        buttonBackspace.setOnClickListener(null);
        buttonEqual.setOnClickListener(null);
    } else if (mCurrentState.equals(CalculatorState.ERROR)) {
        buttonPoint.setOnClickListener(null);
        button0.setOnClickListener(null);
        button1.setOnClickListener(null);
        button2.setOnClickListener(null);
        button3.setOnClickListener(null);
        button4.setOnClickListener(null);
        button5.setOnClickListener(null);
        button6.setOnClickListener(null);
        button7.setOnClickListener(null);
        button8.setOnClickListener(null);
```

```
            button9.setOnClickListener(null);
            buttonA.setOnClickListener(null);
            buttonB.setOnClickListener(null);
            buttonC.setOnClickListener(null);
            buttonD.setOnClickListener(null);
            buttonE.setOnClickListener(null);
            buttonF.setOnClickListener(null);
            buttonPlusMinus.setOnClickListener(null);
            buttonPlus.setOnClickListener(null);
            buttonMinus.setOnClickListener(null);
            buttonMultiply.setOnClickListener(null);
            buttonDivide.setOnClickListener(null);
            buttonEqual.setOnClickListener(this);
        }
    }
}

protected NumberPolarity getPolarity(){ return currentPolarity; }

protected void setPolarity(NumberPolarity polarity){
    if (currentPolarity != polarity) {
        currentPolarity = polarity;
    }
}

protected String setBase(String currentBase, String base){
    if (currentBase != base) {
        currentBase = base;
    }
    hasBase = true;

    return currentBase;
}

protected void insert(String text) {
    if (mCurrentState.equals(CalculatorState.READY)) {
        editTextInput.setText(text);
        editTextInput.setSelection(editTextInput.getText().length());
    } else if (mCurrentState.equals(CalculatorState.INPUT)) {
        editTextInput.setText(editTextInput.getText() + text);
        editTextInput.setSelection(editTextInput.getText().length());
    }
}

protected void onDelete() {
    backspace();
    editTextInput.setSelection(editTextInput.getText().length());
}
```

```
protected void onClear() {
   setState(CalculatorState.READY);
   setPolarity(NumberPolarity.POSITIVE);
   expressionCarrier = new SpannableStringBuilder("");
   formatedInput = new SpannableStringBuilder("");
   numericExpression = new SpannableStringBuilder("");
   numInStack = 0;
   fullNumStack = false;
   currentSign = "+";
   result = 0d;
   editTextInput.setText("0");
   hasBase = false;
   mCurrentError = ErrorHandler.ErrorType.NULL;
   Result.editText_dec_result.setText(null);
   Result.editText_hex_result.setText(null);
   Result.editText_bin_result.setText(null);
   editTextExpession.setText(numericExpression.toString());
   editTextExpession.setSelection(editTextExpession.getText().length());
}

protected void onNumberReset() {
   editTextInput.setText("0");
   hasBase = false;
}

protected void backspace() {
   String input = editTextInput.getText().toString();
   String newText;

   if (hasBase) {
      setState(CalculatorState.READY);
      onNumberReset();
   } else {
      try {
         if (input.length() > 0) {
            if (input.length() == 1) {
               onNumberReset();
               setState(CalculatorState.READY);
            } else {
               newText = input.substring(0, input.length() - 1);
               editTextInput.setText(newText);
               if (newText.equals("0")) {//Objects.equals(newText,"0")
                  setState(CalculatorState.READY);
               } else {
                  setState(CalculatorState.INPUT);
               }
            }
         }
      } catch (Exception e) {
```

```
        }
      }
    }

    protected void convertInput(String clearedInput, String base) {
      switch (base) {
        case "10":
          convertedInput = clearedInput;
          break;
        case "16":
          if (clearedInput.contains(".")) {
            String hexformated = "0x" + clearedInput + "p0";
            convertedInput = BaseConvertion.FromHexToDecimalWithFraction(hexformated);
          } else {
            convertedInput = BaseConvertion.FromHexToDecimal(clearedInput);
          }
          break;
        case "2":
          if (clearedInput.contains(".")) {
            convertedInput =
BaseConvertion.FromBinaryToDecimalWithFraction(clearedInput);
          } else {
            convertedInput = BaseConvertion.FromBinaryToDecimal(clearedInput);
          }
          break;
      }
    }

    protected void toDecimal(String input) {

      if(input.length() > maxLengthforDec) {
        ErrorHandler.echoError(ErrorHandler.ErrorType.LENGTH, this.getActivity());
      } else {
        if(input.endsWith(".")) {
          input = input.substring(0, input.length() - 1);
        }
        String clearInput, polarityInput;
        String base = "10";

        if(hasBase) {
          clearInput = BaseFormater.clearBase(input, currentBase);

          if (BaseFormater.isDecimalFast(clearInput)) {
            currentBase = setBase(currentBase, base);
            convertInput(clearInput, currentBase);
            polarityInput = clearInput;
            formatedInput.clear();//check
            formatedInput = BaseFormater.format(polarityInput,currentBase);
            editTextInput.setText(formatedInput);
```

```
            } else {
                ErrorHandler.echoError(ErrorHandler.ErrorType.BASE, this.getActivity());
                if(hasBase) {
                    setState(CalculatorState.EVALUATE);
                } else {
                    setState(CalculatorState.INPUT);
                }
            }
        } else {
            if (BaseFormater.isDecimalFast(input)) {
                currentBase = setBase(currentBase,base);
                convertInput(input, currentBase);
                polarityInput = input;
                formatedInput.clear();
                formatedInput = BaseFormater.format(polarityInput,currentBase);
                editTextInput.setText(formatedInput);
            } else {
                ErrorHandler.echoError(ErrorHandler.ErrorType.BASE, this.getActivity());
                if (hasBase) {
                    setState(CalculatorState.EVALUATE);
                } else {
                    setState(CalculatorState.INPUT);
                }
            }
        }
    }
}

protected void toHex() {
    String input = editTextInput.getText().toString();

    if (input.length() > maxLengthforHex) {
        ErrorHandler.echoError(ErrorHandler.ErrorType.LENGTH, this.getActivity());
    } else {
        if (input.endsWith(".")) {
            input = input.substring(0, input.length() - 1);
        }
        String clearInput, polarityInput;
        String base = "16";

        if (hasBase) {
            clearInput = input.substring(0, input.length() - currentBase.length());

            if (BaseFormater.isHexFast(clearInput)) {
                currentBase = setBase(currentBase, base);
                convertInput(clearInput, currentBase);
                polarityInput = clearInput;
                formatedInput.clear();
                formatedInput = BaseFormater.format(polarityInput,currentBase);
```

```
            editTextInput.setText(formatedInput);
        } else {
            ErrorHandler.echoError(ErrorHandler.ErrorType.BASE, this.getActivity());
            if (hasBase) {
                setState(CalculatorState.EVALUATE);
            } else {
                setState(CalculatorState.INPUT);
            }
        }
    } else {
        if (BaseFormater.isHexFast(input)) {
            currentBase = setBase(currentBase,base);
            convertInput(input, currentBase);
            polarityInput = input;
            formatedInput.clear();
            formatedInput = BaseFormater.format(polarityInput,currentBase);
            editTextInput.setText(formatedInput);
        } else {
            ErrorHandler.echoError(ErrorHandler.ErrorType.BASE, this.getActivity());
            if (hasBase) {
                setState(CalculatorState.EVALUATE);
            } else {
                setState(CalculatorState.INPUT);
            }
        }
    }
}

protected void toBin() {
    String input = editTextInput.getText().toString();

    if (input.length() > maxLengthforBin) {
        ErrorHandler.echoError(ErrorHandler.ErrorType.LENGTH, this.getActivity());
    } else {
        if (input.endsWith(".")) {
            input = input.substring(0, input.length() - 1);
        }
        String clearInput, polarityInput;
        String base = "2";

        if (hasBase) {
            clearInput = input.substring(0, input.length() - currentBase.length());

            if (BaseFormater.isBinFast(clearInput)) {
                currentBase = setBase(currentBase,base);
                convertInput(clearInput, currentBase);
                polarityInput = clearInput;
                formatedInput.clear();
```

```
            formatedInput = BaseFormater.format(polarityInput,currentBase);
            editTextInput.setText(formatedInput);
         } else {
            ErrorHandler.echoError(ErrorHandler.ErrorType.BASE, this.getActivity());
            if (hasBase) {
               setState(CalculatorState.EVALUATE);
            } else {
               setState(CalculatorState.INPUT);
            }
         }
      } else {
         if (BaseFormater.isBinFast(input)) {
            currentBase = setBase(currentBase,base);
            convertInput(input, currentBase);
            polarityInput = input;
            formatedInput.clear();
            formatedInput = BaseFormater.format(polarityInput,currentBase);
            editTextInput.setText(formatedInput);
         } else {
            ErrorHandler.echoError(ErrorHandler.ErrorType.BASE, this.getActivity());
            if (hasBase) {
               setState(CalculatorState.EVALUATE);
            } else {
               setState(CalculatorState.INPUT);
            }
         }
      }
   }
}

protected void cutter() {
   numInStack++;
   if (numInStack >= 3) {
      fullNumStack = true;
      mCurrentError = ErrorHandler.ErrorType.FULLSTACK;
   } else {
      if (numInStack == 1) {
         supremBase = currentBase;
      }
      fullNumStack = false;
   }
}

protected void pushToNumericExpression(SpannableStringBuilder input, InputType type) {
   if (type == InputType.NUMBER){
      cutter();
   }
   if ((fullNumStack) && (getState() != CalculatorState.RESULT)) {
      if (mCurrentError.equals(ErrorHandler.ErrorType.FULLSTACK)) {
```

```
            ErrorHandler.echoError(mCurrentError, this.getActivity());
            setState(CalculatorState.ERROR);
        }
    } else {
        numericExpression.append(input);
        editTextExpession.setText(numericExpression);
        editTextExpession.setSelection(editTextExpession.getText().length());
    }
}

protected void pushToResult(String pastSign) {
    switch(pastSign) {
        case "+":
            try {
                result = result + Double.parseDouble(convertedInput);
                processResult(result);
            } catch(NumberFormatException|ArithmeticException numberEx) {
                System.out.print(numberEx);
            }
            break;
        case "-":
            try {
                result = result - Double.parseDouble(convertedInput);
                processResult(result);
            } catch(NumberFormatException|ArithmeticException numberEx) {
                System.out.print(numberEx);
            }
            break;
        case "*":
            try {
                result = result * Double.parseDouble(convertedInput);
                processResult(result);
            } catch(NumberFormatException|ArithmeticException numberEx) {
                System.out.print(numberEx);
            }
            break;
        case "/":
            try {
                if (Double.parseDouble(convertedInput) != 0) {
                    result = result / Double.parseDouble(convertedInput);
                    processResult(result);
                } else {
                    editTextInput.setText(getResources().getString(R.string.dividebyzero));
                    mCurrentError = ErrorHandler.ErrorType.DIVIDEWITHZERO;
                    setState(CalculatorState.ERROR);
                }
            } catch(NumberFormatException|ArithmeticException ae) {
                System.out.print(ae);
            }
```

```
            break;
        }
    }

    protected void processResult(Double result) {
        String mResult = uselessZero(String.valueOf(result));
        if (result > 0) {
            if (mResult.contains(".")) {

Result.editText_hex_result.setText(BaseConvertion.FromDecimalToHexWithFraction(mResul
t).toString());

Result.editText_hex_result.setSelection(Result.editText_hex_result.getText().length());


Result.editText_bin_result.setText(BaseConvertion.FromDecimalToBinaryWithFraction(mRes
ult).toString());

Result.editText_bin_result.setSelection(Result.editText_bin_result.getText().length());
            } else {
                Result.editText_hex_result.setText(BaseConvertion.FromDecimalToHex(mResult));

Result.editText_hex_result.setSelection(Result.editText_hex_result.getText().length());


Result.editText_bin_result.setText(BaseConvertion.FromDecimalToBinary(mResult));

Result.editText_bin_result.setSelection(Result.editText_bin_result.getText().length());
            }

            Result.editText_dec_result.setText(mResult);
            Result.editText_dec_result.setSelection(Result.editText_dec_result.getText().length());

        } else if(result < 0) {
            String clearPolarity, tDec, cBinary, bResult, hResult;
            clearPolarity = mResult.replaceAll("[-]","");
            if (mResult.contains(".")) {
                cBinary =
BaseConvertion.FromDecimalToBinaryWithFraction(clearPolarity).toString();

                bResult = "-" + cBinary;
                tDec = BaseConvertion.FromBinaryToDecimalWithFraction(cBinary);

                StringBuilder hexResult = BaseConvertion.FromDecimalToHexWithFraction(tDec);

                hResult = "-" + hexResult.toString().toUpperCase();

                Result.editText_hex_result.setText(hResult);
```

```
Result.editText_hex_result.setSelection(Result.editText_hex_result.getText().length());

            Result.editText_bin_result.setText(bResult);

Result.editText_bin_result.setSelection(Result.editText_bin_result.getText().length());
        } else {
            cBinary = BaseConvertion.FromDecimalToBinary(clearPolarity);

            bResult = "-" + cBinary;
            tDec = BaseConvertion.FromBinaryToDecimal(cBinary);
            hResult = "-" + BaseConvertion.FromDecimalToHex(tDec).toUpperCase();

            Result.editText_hex_result.setText(hResult);

Result.editText_hex_result.setSelection(Result.editText_hex_result.getText().length());

            Result.editText_bin_result.setText(bResult);

Result.editText_bin_result.setSelection(Result.editText_bin_result.getText().length());
        }
        Result.editText_dec_result.setText(mResult);
        Result.editText_dec_result.setSelection(Result.editText_dec_result.getText().length());
    }
    else if (result == 0) {
        Result.editText_dec_result.setText(mResult);
        Result.editText_dec_result.setSelection(Result.editText_dec_result.getText().length());

        Result.editText_hex_result.setText(mResult);
        Result.editText_hex_result.setSelection(Result.editText_hex_result.getText().length());

        Result.editText_bin_result.setText(mResult);
        Result.editText_bin_result.setSelection(Result.editText_bin_result.getText().length());
    }
    else {
        setState(CalculatorState.ERROR);
    }
  }

  protected void result() {
    if (mCurrentError == ErrorHandler.ErrorType.DIVIDEWITHZERO) {
        ErrorHandler.echoError(mCurrentError, this.getActivity());
        onClear();
    } else {
        setState(CalculatorState.RESULT);
        pushToNumericExpression(expressionCarrier.append(" "), InputType.SPACE);
        pushToNumericExpression(formatedInput, InputType.NUMBER);
        expressionCarrier.clear();
        currentSign = null;
        numInStack = 0;
```

```
        fullNumStack = false;
        mCurrentError = ErrorHandler.ErrorType.NULL;
        numericExpression.append("=");
        editTextExpession.setText(numericExpression);
        editTextExpession.setSelection(editTextExpession.getText().length());
        switch (supremBase) {
          case "10":

editTextInput.setText(BaseFormater.format(Result.editText_dec_result.getText().toString(),
supremBase));

expressionCarrier.append(BaseFormater.format(Result.editText_dec_result.getText().toStrin
g(), supremBase));
              break;
          case  "16":

editTextInput.setText(BaseFormater.format(Result.editText_hex_result.getText().toString(),s
upremBase));

expressionCarrier.append(BaseFormater.format(Result.editText_hex_result.getText().toStrin
g(),supremBase));
              break;
          case "2":

editTextInput.setText(BaseFormater.format(Result.editText_bin_result.getText().toString(),
supremBase));

expressionCarrier.append(BaseFormater.format(Result.editText_bin_result.getText().toStrin
g(), supremBase));
              break;
        }
        currentBase = setBase(currentBase, supremBase);
        convertInput(BaseFormater.clearBase(editTextInput.getText().toString(),
currentBase), currentBase);
      }
   }

   protected void plus_minus(String base) {
      String input = editTextInput.getText().toString();
      switch (base) {
        case "10":
          PlusMinus.DecimalPolarity(currentPolarity,input);
          break;
        case "16":
          PlusMinus.HexPolarity(currentPolarity,input);
          break;
        case "2":
          PlusMinus.BinaryPolarity(currentPolarity,input);
          break;
```

```
      }
      if (getState() == CalculatorState.RESULT) {
         expressionCarrier.clear();

expressionCarrier.append(BaseFormater.format(BaseFormater.clearBase(editTextInput.getT
ext().toString(), supremBase), supremBase));
         result *= -1;
      }
   }

   protected void plus(String sign) {
      String lastSign;
      if (fullNumStack) {
         ErrorHandler.echoError(ErrorHandler.ErrorType.FULLSTACK, this.getActivity());
         setState(CalculatorState.ERROR);
      } else {
         if ((sign == null) && (mCurrentState.equals(CalculatorState.READY) ||
mCurrentState.equals(CalculatorState.RESULT)) && (numericExpression.length() > 0)) {
            numericExpression = new SpannableStringBuilder("");
            numInStack++;
            editTextInput.setText("0");
            currentSign = "+";
            pushToNumericExpression(expressionCarrier.append(" "+currentSign),
InputType.SIGN);
            expressionCarrier.clear();
         } else if((sign == "+") && (mCurrentState.equals(CalculatorState.EVALUATE))) {
            currentSign = "+";
            pushToNumericExpression(expressionCarrier.append(" "), InputType.SPACE);
            pushToNumericExpression(formatedInput, InputType.NUMBER);
            pushToNumericExpression(expressionCarrier.append(currentSign),
InputType.SIGN);
            expressionCarrier.clear();
         } else if((sign == "+") && (mCurrentState.equals(CalculatorState.READY))) {
            String ttt = "imhere";
         } else if(sign != "+") {
            if ((numericExpression.length() > 0) &&
(mCurrentState.equals(CalculatorState.READY))) {
               currentSign = "+";
               char last = numericExpression.charAt(numericExpression.length() - 1);
               lastSign = Character.toString(last);
               if (currentSign != lastSign) {
                  numericExpression.delete(numericExpression.length() - 1,
numericExpression.length());
                  pushToNumericExpression(expressionCarrier.append(currentSign),
InputType.SIGN);
                  expressionCarrier.clear();
               }
            } else if ((numericExpression.length() == 0) &&
(mCurrentState.equals(CalculatorState.READY))) {
```

```
            String testt = "im hrere";
          } else {
            currentSign = "+";
            pushToNumericExpression(expressionCarrier.append(" "),InputType.SPACE);
            pushToNumericExpression(formatedInput, InputType.NUMBER);
            pushToNumericExpression(expressionCarrier.append(currentSign),
InputType.SIGN);
            expressionCarrier.clear();
          }
        }
      }
    }

    protected void minus(String sign) {
      String lastSign;
      if (fullNumStack) {
        ErrorHandler.echoError(ErrorHandler.ErrorType.FULLSTACK, this.getActivity());
      } else {
        if ((sign == null) && (mCurrentState.equals(CalculatorState.READY) ||
mCurrentState.equals(CalculatorState.RESULT)) && (numericExpression.length() > 0)) {
          numericExpression = new SpannableStringBuilder("");
          numInStack++;
          editTextInput.setText("0");
          currentSign = "-";
          pushToNumericExpression(expressionCarrier.append(" "+currentSign),
InputType.SIGN);
          expressionCarrier.clear();
        } else if ((sign == "-") && (mCurrentState.equals(CalculatorState.EVALUATE))) {
          currentSign = "-";
          pushToNumericExpression(expressionCarrier.append(" "), InputType.SPACE);
          pushToNumericExpression(formatedInput,InputType.NUMBER);
          pushToNumericExpression(expressionCarrier.append(currentSign),InputType.SIGN);
          expressionCarrier.clear();
          //resetPolarity();
        } else if((sign == "-") && (mCurrentState.equals(CalculatorState.READY))) {
          String ttt = "imhere";
        } else if(sign != "-") {
          if ((numericExpression.length() > 0) &&
(mCurrentState.equals(CalculatorState.READY))) {
            currentSign = "-";
            char last = numericExpression.charAt(numericExpression.length() - 1);
            lastSign = Character.toString(last);
            if (currentSign != lastSign) {
              numericExpression.delete(numericExpression.length() -
1,numericExpression.length());

pushToNumericExpression(expressionCarrier.append(currentSign),InputType.SIGN);
              expressionCarrier.clear();
            }
```

```
            } else if ((numericExpression.length() == 0) &&
(mCurrentState.equals(CalculatorState.READY))) {
                String testt = "im hrere";
            } else {
                currentSign = "-";
                pushToNumericExpression(expressionCarrier.append(" "),InputType.SPACE);
                pushToNumericExpression(formatedInput, InputType.NUMBER);

pushToNumericExpression(expressionCarrier.append(currentSign),InputType.SIGN);
                expressionCarrier.clear();
            }
        }
    }
  }

  protected void multiply(String sign) {
    String lastSign;
    if (fullNumStack) {
      ErrorHandler.echoError(ErrorHandler.ErrorType.FULLSTACK, this.getActivity());
    } else {
      if ((sign == null) && (mCurrentState.equals(CalculatorState.READY) ||
mCurrentState.equals(CalculatorState.RESULT)) && (numericExpression.length() > 0)) {
        numericExpression = new SpannableStringBuilder("");
        numInStack++;
        editTextInput.setText("0");
        currentSign = "*";
        pushToNumericExpression(expressionCarrier.append(" "+currentSign),
InputType.SIGN);
        expressionCarrier.clear();
      } else if ((sign == "*") && (mCurrentState.equals(CalculatorState.EVALUATE))) {
        currentSign = "*";
        pushToNumericExpression(expressionCarrier.append(" "),InputType.SPACE);
        pushToNumericExpression(formatedInput, InputType.NUMBER);
        pushToNumericExpression(expressionCarrier.append(currentSign),InputType.SIGN);
        expressionCarrier.clear();
      } else if ((sign == "*") && (mCurrentState.equals(CalculatorState.READY))) {
        String ttt = "imhere";
      } else if(sign != "*") {
        if ((numericExpression.length() > 0) &&
(mCurrentState.equals(CalculatorState.READY))) {
          currentSign = "*";
          char last = numericExpression.charAt(numericExpression.length() - 1);
          lastSign = Character.toString(last);
          if (currentSign != lastSign) {
            numericExpression.delete(numericExpression.length() - 1,
numericExpression.length());
            pushToNumericExpression(expressionCarrier.append(currentSign),
InputType.SIGN);
            expressionCarrier.clear();
```

```
            }
          } else if ((numericExpression.length() == 0) &&
(mCurrentState.equals(CalculatorState.READY))) {
            String testt = "im hrere";
          } else {
            currentSign = "*";
            pushToNumericExpression(expressionCarrier.append(" "),InputType.SPACE);
            pushToNumericExpression(formatedInput,InputType.NUMBER);

pushToNumericExpression(expressionCarrier.append(currentSign),InputType.SIGN);
            expressionCarrier.clear();
          }
        }
      }
    }

    protected void divide(String sign) {
      String lastSign;
      if (fullNumStack) {
        ErrorHandler.echoError(ErrorHandler.ErrorType.FULLSTACK, this.getActivity());
      } else {
        if ((sign == null) && (mCurrentState.equals(CalculatorState.READY) ||
mCurrentState.equals(CalculatorState.RESULT)) && (numericExpression.length() > 0)) {
          numericExpression = new SpannableStringBuilder("");
          numInStack++;
          editTextInput.setText("0");
          currentSign = "/";
          pushToNumericExpression(expressionCarrier.append(" "+currentSign),
InputType.SIGN);
          expressionCarrier.clear();
        } else if ((sign == "/") && (mCurrentState.equals(CalculatorState.EVALUATE))) {
          currentSign = "/";
          pushToNumericExpression(expressionCarrier.append(" "),InputType.SPACE);
          pushToNumericExpression(formatedInput,InputType.NUMBER);
          pushToNumericExpression(expressionCarrier.append(currentSign),InputType.SIGN);
          expressionCarrier.clear();
        } else if ((sign == "/") && (mCurrentState.equals(CalculatorState.READY))) {
          String ttt = "imhere";
        } else if (sign != "/") {
          if ((numericExpression.length() > 0) &&
(mCurrentState.equals(CalculatorState.READY))) {
            currentSign = "/";
            char last = numericExpression.charAt(numericExpression.length() - 1);
            lastSign = Character.toString(last);
            if (currentSign != lastSign) {
              numericExpression.delete(numericExpression.length() - 1,
numericExpression.length());
              pushToNumericExpression(expressionCarrier.append(currentSign),
InputType.SIGN);
```

```
                expressionCarrier.clear();
            }
        } else if ((numericExpression.length() == 0) &&
(mCurrentState.equals(CalculatorState.READY))) {
            String testt = "im hrere";
        } else {
            currentSign = "/";
            pushToNumericExpression(expressionCarrier.append(" "),InputType.SPACE);
            pushToNumericExpression(formatedInput, InputType.NUMBER);
            pushToNumericExpression(expressionCarrier.append(currentSign),
InputType.SIGN);
            expressionCarrier.clear();
        }
      }
    }
  }

  protected String uselessZero(String input) {
    StringBuilder deleteUselessZeros = new StringBuilder("");
    deleteUselessZeros.append(input);
    for (int i=deleteUselessZeros.length(); i>0; i--) {
      if (deleteUselessZeros.charAt(deleteUselessZeros.length() - 1) == '0') {
        deleteUselessZeros.setLength(deleteUselessZeros.length()-1);
      }
    }
    if (deleteUselessZeros.charAt(deleteUselessZeros.length() - 1) == '.') {
      deleteUselessZeros.setLength(deleteUselessZeros.length()-1);
    }
    return deleteUselessZeros.toString();
  }

  private void initialSetup() {

    setState(CalculatorState.READY);
    setPolarity(NumberPolarity.POSITIVE);
    mCurrentError = ErrorHandler.ErrorType.NULL;

    numericExpression = new SpannableStringBuilder("");
    expressionCarrier = new SpannableStringBuilder("");
    formatedInput = new SpannableStringBuilder("");

    hasBase = false;
    fullNumStack = false;
    numInStack = 0;

    maxLengthforDec = 10;
    maxLengthforHex = 8;
    maxLengthforBin = 32;
  }
```

```
    private enum CalculatorState {
        READY, INPUT, EVALUATE, RESULT, ERROR
    }

    public enum NumberPolarity {
        POSITIVE, NEGATIVE
    }

    private enum InputType {
        NUMBER, SIGN, SPACE
    }
}
```

CONVERTER.JAVA

```
package com.calcobin;

import android.app.Activity;
import android.support.v4.app.Fragment;
import android.content.Context;
import android.os.Bundle;
import android.text.Editable;
import android.text.InputFilter;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.calcobin.helpers.BaseConvertion;
import com.calcobin.helpers.BaseFormater;
import com.calcobin.helpers.ErrorHandler;


public class Converter extends Fragment implements View.OnClickListener{

   EditText textDec, textHex, textBin, textAscii;

   Button buttonDecimal, buttonHexadecimal, buttonBinary, buttonAscii, button0, button1,
button2, button3, button4, button5, button6, button7, button8, button9,
         buttonA, buttonB, buttonC, buttonD, buttonE, buttonF, buttonDot, buttonClear,
buttonDelete;

   private BaseState mCurrentBase;

   private ConverterState mCurrentState;

   private ErrorHandler.ErrorType mCurrentError;

   private int maxLengthforDec = 10, maxLengthforHex = 8, maxLengthforBin = 32, limit;

   private boolean fullstack;

   private Activity activity;


   private TextWatcher decTextWatcher = new TextWatcher() {
      @Override
```

```
public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {

   buttonDelete.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         delete(textDec);
      }
   });
   buttonClear.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         clear();
      }
   });

}

@Override
public void afterTextChanged(Editable s) {

   String decInput = s.toString();

   if(decInput.length() == 0){
      mCurrentError = ErrorHandler.ErrorType.NULL;
      textBin.setText("");
      textHex.setText("");
      textAscii.setText("");
      buttonDot.setEnabled(true);

   }else if(decInput.length() > maxLengthforDec){
      ErrorHandler.echoError(ErrorHandler.ErrorType.DEC_LENGTH, activity);
      textDec.setSelection(textDec.getText().length());

   }else if(decInput.endsWith(".")) {

      if(BaseFormater.isDecimalFast(decInput)){
         buttonDot.setEnabled(false);

         String[] splits = decInput.split("\\.");
         String beforeDot = splits[0];

         String hexResult = BaseConvertion.FromDecimalToHex(beforeDot);
         String binaryResult = BaseConvertion.FromDecimalToBinary(beforeDot);
         StringBuilder asciiResult = BaseConvertion.FromDecimalToAscii(beforeDot);

         textDec.setSelection(textDec.getText().length());
```

```
            textBin.setText(binaryResult);
            textBin.setSelection(textBin.getText().length());

            textHex.setText(hexResult);
            textHex.setSelection(textHex.getText().length());

            textAscii.setText(asciiResult.toString());
          }else{
            setState(ConverterState.ERROR);
          }
        }else{

          if (decInput.contains(".")){

            if(BaseFormater.isDecimalFast(decInput)){
              buttonDot.setEnabled(false);

              try {

                String[] splits = decInput.split("\\.");
                String beforeDot = splits[0];
                String afterDot = splits[1];

                if(Double.parseDouble(afterDot)== 0){

                  String hexResult = BaseConvertion.FromDecimalToHex(beforeDot);
                  String binaryResult = BaseConvertion.FromDecimalToBinary(beforeDot);
                  StringBuilder asciiResult =
BaseConvertion.FromDecimalToAscii(beforeDot);

                  textDec.setSelection(textDec.getText().length());

                  textBin.setText(binaryResult);
                  textBin.setSelection(textBin.getText().length());

                  textHex.setText(hexResult);
                  textHex.setSelection(textHex.getText().length());

                  textAscii.setText(asciiResult.toString());

                }else{

                  textDec.setSelection(textDec.getText().length());

                   /* Convertion from Decimal to Binary with "."  */
                  StringBuilder binResult =
BaseConvertion.FromDecimalToBinaryWithFraction(decInput);
                  textBin.setText(binResult.toString());
```

```
                    textBin.setSelection(textBin.getText().length());
                    /* end of Convertion from Decimal to Binary with "."  */

                    /* Convertion from Decimal to Hex with "."  */
                    StringBuilder hexResult =
BaseConvertion.FromDecimalToHexWithFraction(decInput);
                    textHex.setText(hexResult.toString());
                    textHex.setSelection(textHex.getText().length());
                     /* end of Convertion from Decimal to Hex with "."  */

                  }
                }catch(NumberFormatException numberEx) {
                  System.out.print(numberEx);
                }
              }else{
                setState(ConverterState.ERROR);
              }
            }else{
              buttonDot.setEnabled(true);

              /* Convertion from Decimal to Hex & Binary without "."  */
              if(BaseFormater.isDecimalFast(decInput)){
                try {

                  String hexResult = BaseConvertion.FromDecimalToHex(decInput);
                  String binaryResult = BaseConvertion.FromDecimalToBinary(decInput);
                  StringBuilder asciiResult = BaseConvertion.FromDecimalToAscii(decInput);

                  textDec.setSelection(textDec.getText().length());

                  textBin.setText(binaryResult);
                  textBin.setSelection(textBin.getText().length());

                  textHex.setText(hexResult);
                  textHex.setSelection(textHex.getText().length());

                  textAscii.setText(asciiResult.toString());

                }
                catch(NumberFormatException numberEx) {
                  System.out.print(numberEx);
                }
              }else {
                setState(ConverterState.ERROR);
              }
            }
          }
        }
      };
```

```
TextWatcher hexTextWatcher = new TextWatcher() {
   @Override
   public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

   @Override
   public void onTextChanged(CharSequence s, int start, int before, int count) {

      buttonDelete.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            delete(textHex);
         }
      });

      buttonClear.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            clear();
         }
      });

   }

   @Override
   public void afterTextChanged(Editable s) {

      String hexInput = s.toString();

      if (hexInput.length() == 0) {

         textBin.setText("");
         textDec.setText("");
         textAscii.setText("");

      } else if(hexInput.endsWith(".")) {

         if(BaseFormater.isHexFast(hexInput)){
            buttonDot.setEnabled(false);

            String[] splits = hexInput.split("\\.");
            String beforeDot = splits[0];

            String binaryResult = BaseConvertion.FromHexToBinary(beforeDot);
            String decimalResult = BaseConvertion.FromHexToDecimal(beforeDot);
            StringBuilder asciiResult = BaseConvertion.FromHexToAscii(beforeDot);

            textHex.setSelection(textHex.getText().length());
```

```
            textBin.setText(binaryResult);
            textBin.setSelection(textBin.getText().length());

            textDec.setText(decimalResult);
            textDec.setSelection(textDec.getText().length());

            textAscii.setText(asciiResult.toString());
          }else {
            setState(ConverterState.ERROR);
          }
        }else{

          if(hexInput.contains(".")){

            if(BaseFormater.isHexFast(hexInput)){
              buttonDot.setEnabled(false);

              try{

                String[] splits = hexInput.split("[.]");
                String beforeDot = splits[0];
                String afterDot = splits[1];

                if(afterDot.matches(".*[A-F].*")){

                  textHex.setSelection(textHex.getText().length());

                  /* Convertion from Hexadecimal to Binary using Decimal */
                  String hexformated = "0x"+ hexInput + "p0";
                  String decfromhex = Double.valueOf(hexformated).toString();
                  StringBuilder binaryResult =
BaseConvertion.FromDecimalToBinaryWithFraction(decfromhex);
                  textBin.setText(binaryResult.toString());
                  textBin.setSelection(textBin.getText().length());
                  /* end of Convertion from Hexadecimal to Binary using Decimal */

                  /* Convertion from Hexadecimal to Decimal */
                  //hexformated = "0xA.Ap0";
                  String decResult =
BaseConvertion.FromHexToDecimalWithFraction(hexformated);
                  textDec.setText(decResult);
                  textDec.setSelection(textDec.getText().length());
                  /* end of Convertion from Hexadecimal to Decimal */

                }else if(afterDot.matches(".*[0-9].*")){

                  if(Double.parseDouble(afterDot) == 0){

                    String binaryResult = BaseConvertion.FromHexToBinary(beforeDot);
```

```
                    String decimalResult = BaseConvertion.FromHexToDecimal(beforeDot);
                    StringBuilder asciiResult = BaseConvertion.FromHexToAscii(beforeDot);

                    textHex.setSelection(textHex.getText().length());

                    textBin.setText(binaryResult);
                    textBin.setSelection(textBin.getText().length());

                    textDec.setText(decimalResult);
                    textDec.setSelection(textDec.getText().length());

                    textAscii.setText(asciiResult.toString());

                  }else{

                    textHex.setSelection(textHex.getText().length());

                  /* Convertion from Hexadecimal to Binary using Decimal with Fraction */
                    String hexformated = "0x"+ hexInput + "p0";
                    String decfromhex = Double.valueOf(hexformated).toString();
                    StringBuilder binaryResult =
BaseConvertion.FromDecimalToBinaryWithFraction(decfromhex);
                    textBin.setText(binaryResult.toString());
                    textBin.setSelection(textBin.getText().length());
                  /* end of Convertion from Hexadecimal to Binary using Decimal */

                  /* Convertion from Hexadecimal to Decimal */
                    //hexformated = "0xA.Ap0";
                    String decResult =
BaseConvertion.FromHexToDecimalWithFraction(hexformated);
                    textDec.setText(decResult);
                    textDec.setSelection(textDec.getText().length());
                  /* end of Convertion from Hexadecimal to Decimal */

                  }
                }

              }
              catch (NumberFormatException numberEx){
                System.out.print(numberEx);
              }
            }else {
              setState(ConverterState.ERROR);
            }
          }else{

            if(BaseFormater.isHexFast(hexInput)){
              buttonDot.setEnabled(true);
```

```
            try {

                String binaryResult = BaseConvertion.FromHexToBinary(hexInput);
                String decimalResult = BaseConvertion.FromHexToDecimal(hexInput);
                StringBuilder asciiResult = BaseConvertion.FromHexToAscii(hexInput);

                textHex.setSelection(textHex.getText().length());

                textBin.setText(binaryResult);
                textBin.setSelection(textBin.getText().length());

                textDec.setText(decimalResult);
                textDec.setSelection(textDec.getText().length());

                textAscii.setText(asciiResult.toString());

            }
            catch(NumberFormatException numberEx) {
                System.out.print(numberEx);
            }
          }else {
            setState(ConverterState.ERROR);
          }
        }
      }
    }
};

TextWatcher binTextWatcher = new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {

      buttonDelete.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
          delete(textBin);
        }
      });

      buttonClear.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
          clear();
        }
      });
```

```
}

@Override
public void afterTextChanged(Editable s) {

   String binaryInput = s.toString();
   if(binaryInput.length() == 0){

      textDec.setText("");
      textHex.setText("");
      textAscii.setText("");

   }else if(binaryInput.length() > maxLengthforBin){
      ErrorHandler.echoError(ErrorHandler.ErrorType.BIN_LENGTH, activity);
      textBin.setSelection(textBin.getText().length());

   }else if(binaryInput.endsWith(".")){

      if(BaseFormater.isBinFast(binaryInput)){
         buttonDot.setEnabled(false);

         String[] splits = binaryInput.split("\\.");
         String beforeDot = splits[0];

         String decResult = BaseConvertion.FromBinaryToDecimal(beforeDot);
         String hexResult = BaseConvertion.FromBinaryToHex(beforeDot);
         StringBuilder asciiResult = BaseConvertion.FromBinaryToAscii(beforeDot);

         textBin.setSelection(textBin.getText().length());

         textDec.setText(decResult);
         textDec.setSelection(textDec.getText().length());

         textHex.setText(hexResult);
         textHex.setSelection(textHex.getText().length());

         textAscii.setText(asciiResult.toString());

      }else {
         setState(ConverterState.ERROR);
      }
   }else{

      if(binaryInput.contains(".")){

         if(BaseFormater.isBinFast(binaryInput)){
            buttonDot.setEnabled(false);

            try{
```

```
                String[] splits = binaryInput.split("\\.");
                String beforeDot = splits[0];
                String afterDot = splits[1];

                if(Double.parseDouble(afterDot)== 0){

                   String decResult = BaseConvertion.FromBinaryToDecimal(beforeDot);
                   String hexResult = BaseConvertion.FromBinaryToHex(beforeDot);
                   StringBuilder asciiResult = BaseConvertion.FromBinaryToAscii(beforeDot);

                   textBin.setSelection(textBin.getText().length());

                   textDec.setText(decResult);
                   textDec.setSelection(textDec.getText().length());

                   textHex.setText(hexResult);
                   textHex.setSelection(textHex.getText().length());

                   textAscii.setText(asciiResult.toString());

                }else{

                   textBin.setSelection(textBin.getText().length());

                   /* Convertion from Binary to Decimal with "."  */
                   String decResult =
BaseConvertion.FromBinaryToDecimalWithFraction(binaryInput);
                   textDec.setText(decResult);
                   textDec.setSelection(textDec.getText().length());
                   /* end of Convertion from Binary to Decimal with "."  */

                   /* Convertion from Binary to Hex with "." using Decimal result  */
                   StringBuilder hexResult =
BaseConvertion.FromDecimalToHexWithFraction(decResult);
                   textHex.setText(hexResult.toString());
                   textHex.setSelection(textHex.getText().length());
                   /* end of Convertion from Binary to Hex with "." using Decimal result */

                }


             }catch(NumberFormatException numberEx) {
                System.out.print(numberEx);
             }

          }else {
             setState(ConverterState.ERROR);
          }
```

```
        }
        else{

           if(BaseFormater.isBinFast(binaryInput)){
              buttonDot.setEnabled(true);
              try {

                 String decResult = BaseConvertion.FromBinaryToDecimal(binaryInput);
                 String hexResult = BaseConvertion.FromBinaryToHex(binaryInput);
                 StringBuilder asciiResult = BaseConvertion.FromBinaryToAscii(binaryInput);

                 textBin.setSelection(textBin.getText().length());

                 textDec.setText(decResult);
                 textDec.setSelection(textDec.getText().length());

                 textHex.setText(hexResult);
                 textHex.setSelection(textHex.getText().length());

                 textAscii.setText(asciiResult.toString());
              }
              catch(NumberFormatException numberEx) {
                 System.out.print(numberEx);
              }
           }else {
              setState(ConverterState.ERROR);
           }
        }
     }

  }
};


TextWatcher asciiTextWatcher = new TextWatcher() {
   @Override
   public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

   @Override
   public void onTextChanged(CharSequence s, int start, int before, int count) {}

   @Override
   public void afterTextChanged(Editable s) {}
};


@Override
public void onPause() {
   super.onPause();
```

```
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        setRetainInstance(true);
        View v = inflater.inflate(R.layout.activity_converter, container, false);

        buttonDecimal = (Button) v.findViewById(R.id.buttonDec);
        buttonHexadecimal = (Button) v.findViewById(R.id.buttonHex);
        buttonBinary = (Button) v.findViewById(R.id.buttonBin);

        buttonDecimal.setBackgroundResource(R.drawable.blue_button);

        buttonAscii = (Button) v.findViewById(R.id.buttonAscii);

        buttonAscii.setTextColor(getResources().getColor(R.color.disable_blue));

        buttonAscii.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Context context = v.getContext();
                int duration = Toast.LENGTH_SHORT;
                Toast toast = Toast.makeText(context,
getResources().getString(R.string.ascii_message), duration);
                toast.show();
            }
        });

        buttonDot = (Button) v.findViewById(R.id.dot);
        buttonClear = (Button) v.findViewById(R.id.clear);
        buttonDelete = (Button) v.findViewById(R.id.delete);

        button0 = (Button) v.findViewById(R.id.zero);
        button1 = (Button) v.findViewById(R.id.one);
        button2 = (Button) v.findViewById(R.id.two);
        button3 = (Button) v.findViewById(R.id.three);
        button4 = (Button) v.findViewById(R.id.four);
        button5 = (Button) v.findViewById(R.id.five);
        button6 = (Button) v.findViewById(R.id.six);
        button7 = (Button) v.findViewById(R.id.seven);
        button8 = (Button) v.findViewById(R.id.eight);
        button9 = (Button) v.findViewById(R.id.nine);
        buttonA = (Button) v.findViewById(R.id.alpha);
        buttonB = (Button) v.findViewById(R.id.beta);
        buttonC = (Button) v.findViewById(R.id.gamma);
        buttonD = (Button) v.findViewById(R.id.delta);
        buttonE = (Button) v.findViewById(R.id.epsilon);
        buttonF = (Button) v.findViewById(R.id.fi);
```

```
    textDec = (EditText) v.findViewById(R.id.editTextDec);
    textBin = (EditText) v.findViewById(R.id.editTextBin);
    textHex = (EditText) v.findViewById(R.id.editTextHex);
    textAscii = (EditText) v.findViewById(R.id.editTextAscii);

    init();

    return v;

}//end of OnCreateView

public void init(){

    activity = this.getActivity();

    button0.setEnabled(true);
    button0.setTextColor(getResources().getColor(R.color.white));
    button1.setEnabled(true);
    button1.setTextColor(getResources().getColor(R.color.white));
    button2.setEnabled(true);
    button2.setTextColor(getResources().getColor(R.color.white));
    button3.setEnabled(true);
    button3.setTextColor(getResources().getColor(R.color.white));
    button4.setEnabled(true);
    button4.setTextColor(getResources().getColor(R.color.white));
    button5.setEnabled(true);
    button5.setTextColor(getResources().getColor(R.color.white));
    button6.setEnabled(true);
    button6.setTextColor(getResources().getColor(R.color.white));
    button7.setEnabled(true);
    button7.setTextColor(getResources().getColor(R.color.white));
    button8.setEnabled(true);
    button8.setTextColor(getResources().getColor(R.color.white));
    button9.setEnabled(true);
    button9.setTextColor(getResources().getColor(R.color.white));
    buttonA.setEnabled(false);
    buttonA.setTextColor(getResources().getColor(R.color.disable_light_blue));
    buttonB.setEnabled(false);
    buttonB.setTextColor(getResources().getColor(R.color.disable_light_blue));
    buttonC.setEnabled(false);
    buttonC.setTextColor(getResources().getColor(R.color.disable_light_blue));
    buttonD.setEnabled(false);
    buttonD.setTextColor(getResources().getColor(R.color.disable_light_blue));
    buttonE.setEnabled(false);
    buttonE.setTextColor(getResources().getColor(R.color.disable_light_blue));
    buttonF.setEnabled(false);
    buttonF.setTextColor(getResources().getColor(R.color.disable_light_blue));
```

```
    textDec.setText(null);
    textDec.setTextIsSelectable(false);

    textHex.setText(null);
    textHex.setFilters(new InputFilter[] {new InputFilter.AllCaps()});
    textHex.setTextIsSelectable(false);

    textBin.setText(null);
    textBin.setTextIsSelectable(false);

    textAscii.setText(null);
    textAscii.setTextIsSelectable(false);

    textAscii.setEnabled(false);
    textBin.setEnabled(false);
    textHex.setEnabled(false);
    textDec.setEnabled(true);

    textDec.setBackgroundResource(R.drawable.edittextresults);

    textDec.setOnFocusChangeListener(new DecFocusListener());
    textHex.setOnFocusChangeListener(new HexFocusListener());
    textBin.setOnFocusChangeListener(new BinFocusListener());
    setBase(BaseState.DECIMAL);
    setState(ConverterState.READY);
    mCurrentError = ErrorHandler.ErrorType.NULL;
    limit = 0;
    fullstack = false;

    buttonDecimal.setOnClickListener(new DecClickListener());
    buttonHexadecimal.setOnClickListener(new HexClickListener());
    buttonBinary.setOnClickListener(new BinClickListener());
}

public class DecFocusListener implements View.OnFocusChangeListener{
    public void onFocusChange(View v, boolean hasFocus) {

        if (hasFocus) {

            if(!mCurrentState.equals(ConverterState.ERROR)){
                setBase(BaseState.DECIMAL);
                textDec.addTextChangedListener(decTextWatcher);

                button0.setOnClickListener(new KeyboardClickListener(getBase()));
                button1.setOnClickListener(new KeyboardClickListener(getBase()));
                button2.setOnClickListener(new KeyboardClickListener(getBase()));
                button3.setOnClickListener(new KeyboardClickListener(getBase()));
                button4.setOnClickListener(new KeyboardClickListener(getBase()));
                button5.setOnClickListener(new KeyboardClickListener(getBase()));
```

```
            button6.setOnClickListener(new KeyboardClickListener(getBase()));
            button7.setOnClickListener(new KeyboardClickListener(getBase()));
            button8.setOnClickListener(new KeyboardClickListener(getBase()));
            button9.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonDot.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonDelete.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonClear.setOnClickListener(new KeyboardClickListener(getBase()));
        }
    } else {

        textDec.setFocusableInTouchMode(false);
        textDec.removeTextChangedListener(decTextWatcher);
    }
  }
}

public class HexFocusListener implements View.OnFocusChangeListener{
  public void onFocusChange(View v, boolean hasFocus) {
    if (hasFocus) {

        if(!mCurrentState.equals(ConverterState.ERROR)){
            setBase(BaseState.HEXADECIMAL);
            textHex.addTextChangedListener(hexTextWatcher);

            button0.setOnClickListener(new KeyboardClickListener(getBase()));
            button1.setOnClickListener(new KeyboardClickListener(getBase()));
            button2.setOnClickListener(new KeyboardClickListener(getBase()));
            button3.setOnClickListener(new KeyboardClickListener(getBase()));
            button4.setOnClickListener(new KeyboardClickListener(getBase()));
            button5.setOnClickListener(new KeyboardClickListener(getBase()));
            button6.setOnClickListener(new KeyboardClickListener(getBase()));
            button7.setOnClickListener(new KeyboardClickListener(getBase()));
            button8.setOnClickListener(new KeyboardClickListener(getBase()));
            button9.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonA.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonB.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonC.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonD.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonE.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonF.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonDot.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonDelete.setOnClickListener(new KeyboardClickListener(getBase()));
            buttonClear.setOnClickListener(new KeyboardClickListener(getBase()));
        }
    } else {

        textHex.setFocusableInTouchMode(false);
        textHex.removeTextChangedListener(hexTextWatcher);
    }
```

```
    }
}

public class BinFocusListener implements View.OnFocusChangeListener {
    public void onFocusChange(View v, boolean hasFocus) {
        if (hasFocus) {

            if(!mCurrentState.equals(ConverterState.ERROR)){
                setBase(BaseState.BINARY);

                textBin.addTextChangedListener(binTextWatcher);

                button0.setOnClickListener(new KeyboardClickListener(getBase()));
                button1.setOnClickListener(new KeyboardClickListener(getBase()));
                buttonDot.setOnClickListener(new KeyboardClickListener(getBase()));
                buttonDelete.setOnClickListener(new KeyboardClickListener(getBase()));
                buttonClear.setOnClickListener(new KeyboardClickListener(getBase()));
            }
        } else {

            textBin.setFocusableInTouchMode(false);
            textBin.removeTextChangedListener(binTextWatcher);
        }
    }
}

public class DecClickListener implements View.OnClickListener{
    public void onClick(View view){

        if(mCurrentState.equals(ConverterState.ERROR)){
            clear();
            setState(ConverterState.EVALUATE);
        }
        buttonDecimal.setBackgroundResource(R.drawable.blue_button);
        textDec.setBackgroundResource(R.drawable.edittextresults);
        buttonBinary.setBackgroundResource(R.drawable.dark_blue_button);
        textBin.setBackgroundResource(R.drawable.edittextresults);
        buttonHexadecimal.setBackgroundResource(R.drawable.dark_blue_button);
        textHex.setBackgroundResource(R.drawable.edittextresults);

        textBin.setEnabled(false);
        textHex.setEnabled(false);
        textDec.setEnabled(true);

        textBin.setCursorVisible(false);
        textHex.setCursorVisible(false);
        textDec.setCursorVisible(true);

        textHex.clearFocus();
```

```
textHex.setFocusable(false);
textHex.setFocusableInTouchMode(false);


textBin.clearFocus();
textBin.setFocusable(false);
textBin.setFocusableInTouchMode(false);

textAscii.clearFocus();
textAscii.setFocusable(false);
textAscii.setFocusableInTouchMode(false);

textDec.clearFocus();
textDec.setFocusable(true);
textDec.setFocusableInTouchMode(true);
textDec.requestFocus();

int lenghtDec = textDec.getText().toString().length();

if (lenghtDec == maxLengthforDec){
   mCurrentError = ErrorHandler.ErrorType.DEC_LENGTH;
   fullstack = true;
   limit = maxLengthforDec;
} else if(lenghtDec > maxLengthforDec){
   clear();
} else{
   mCurrentError = ErrorHandler.ErrorType.NULL;
   limit = lenghtDec;
   fullstack = false;
}

buttonDot.setEnabled(true);
button0.setEnabled(true);
button0.setTextColor(getResources().getColor(R.color.white));
button1.setEnabled(true);
button1.setTextColor(getResources().getColor(R.color.white));
button2.setEnabled(true);
button2.setTextColor(getResources().getColor(R.color.white));
button3.setEnabled(true);
button3.setTextColor(getResources().getColor(R.color.white));
button4.setEnabled(true);
button4.setTextColor(getResources().getColor(R.color.white));
button5.setEnabled(true);
button5.setTextColor(getResources().getColor(R.color.white));
button6.setEnabled(true);
button6.setTextColor(getResources().getColor(R.color.white));
button7.setEnabled(true);
button7.setTextColor(getResources().getColor(R.color.white));
button8.setEnabled(true);
```

```
        button8.setTextColor(getResources().getColor(R.color.white));
        button9.setEnabled(true);
        button9.setTextColor(getResources().getColor(R.color.white));
        buttonA.setEnabled(false);
        buttonA.setTextColor(getResources().getColor(R.color.disable_light_blue));
        buttonB.setEnabled(false);
        buttonB.setTextColor(getResources().getColor(R.color.disable_light_blue));
        buttonC.setEnabled(false);
        buttonC.setTextColor(getResources().getColor(R.color.disable_light_blue));
        buttonD.setEnabled(false);
        buttonD.setTextColor(getResources().getColor(R.color.disable_light_blue));
        buttonE.setEnabled(false);
        buttonE.setTextColor(getResources().getColor(R.color.disable_light_blue));
        buttonF.setEnabled(false);
        buttonF.setTextColor(getResources().getColor(R.color.disable_light_blue));
    }
}

public class HexClickListener implements View.OnClickListener{
    public void onClick(View view){
        if(mCurrentState.equals(ConverterState.ERROR)){
            clear();
            setState(ConverterState.EVALUATE);
        }
        buttonHexadecimal.setBackgroundResource(R.drawable.blue_button);
        textHex.setBackgroundResource(R.drawable.edittextresults);
        buttonBinary.setBackgroundResource(R.drawable.dark_blue_button);
        textBin.setBackgroundResource(R.drawable.edittextresults);
        buttonDecimal.setBackgroundResource(R.drawable.dark_blue_button);
        textDec.setBackgroundResource(R.drawable.edittextresults);

        textBin.setEnabled(false);
        textDec.setEnabled(false);
        textHex.setEnabled(true);

        textDec.setCursorVisible(false);
        textBin.setCursorVisible(false);
        textHex.setCursorVisible(true);

        textDec.clearFocus();
        textDec.setFocusable(false);
        textDec.setFocusableInTouchMode(false);

        textBin.clearFocus();
        textBin.setFocusable(false);
        textBin.setFocusableInTouchMode(false);

        textAscii.clearFocus();
        textAscii.setFocusable(false);
```

```
textAscii.setFocusableInTouchMode(false);

textHex.clearFocus();
textHex.setFocusable(true);
textHex.setFocusableInTouchMode(true);
textHex.requestFocus();

int lenghtHex = textHex.getText().toString().length();
if(lenghtHex == maxLengthforHex){
   mCurrentError = ErrorHandler.ErrorType.HEX_LENGTH;
   fullstack = true;
   limit = maxLengthforHex;
}else if(lenghtHex > maxLengthforHex){
   clear();
}else{
   mCurrentError = ErrorHandler.ErrorType.NULL;
   limit = lenghtHex;
   fullstack = false;
}

buttonDot.setEnabled(true);
button0.setEnabled(true);
button0.setTextColor(getResources().getColor(R.color.white));
button1.setEnabled(true);
button1.setTextColor(getResources().getColor(R.color.white));
button2.setEnabled(true);
button2.setTextColor(getResources().getColor(R.color.white));
button3.setEnabled(true);
button3.setTextColor(getResources().getColor(R.color.white));
button4.setEnabled(true);
button4.setTextColor(getResources().getColor(R.color.white));
button5.setEnabled(true);
button5.setTextColor(getResources().getColor(R.color.white));
button6.setEnabled(true);
button6.setTextColor(getResources().getColor(R.color.white));
button7.setEnabled(true);
button7.setTextColor(getResources().getColor(R.color.white));
button8.setEnabled(true);
button8.setTextColor(getResources().getColor(R.color.white));
button9.setEnabled(true);
button9.setTextColor(getResources().getColor(R.color.white));
buttonA.setEnabled(true);
buttonA.setTextColor(getResources().getColor(R.color.white));
buttonB.setEnabled(true);
buttonB.setTextColor(getResources().getColor(R.color.white));
buttonC.setEnabled(true);
buttonC.setTextColor(getResources().getColor(R.color.white));
buttonD.setEnabled(true);
buttonD.setTextColor(getResources().getColor(R.color.white));
```

```
        buttonE.setEnabled(true);
        buttonE.setTextColor(getResources().getColor(R.color.white));
        buttonF.setEnabled(true);
        buttonF.setTextColor(getResources().getColor(R.color.white));
    }
}

public class BinClickListener implements View.OnClickListener{
    public void onClick(View view){
        if(mCurrentState.equals(ConverterState.ERROR)){
            clear();
            setState(ConverterState.EVALUATE);
        }
        buttonBinary.setBackgroundResource(R.drawable.blue_button);
        textBin.setBackgroundResource(R.drawable.edittextresults);
        buttonHexadecimal.setBackgroundResource(R.drawable.dark_blue_button);
        textHex.setBackgroundResource(R.drawable.edittextresults);
        buttonDecimal.setBackgroundResource(R.drawable.dark_blue_button);
        textDec.setBackgroundResource(R.drawable.edittextresults);

        textDec.setEnabled(false);
        textHex.setEnabled(false);
        textBin.setEnabled(true);

        textDec.setCursorVisible(false);
        textHex.setCursorVisible(false);
        textBin.setCursorVisible(true);

        textDec.clearFocus();
        textDec.setFocusable(false);
        textDec.setFocusableInTouchMode(false);

        textHex.clearFocus();
        textHex.setFocusable(false);
        textHex.setFocusableInTouchMode(false);

        textAscii.clearFocus();
        textAscii.setFocusable(false);
        textAscii.setFocusableInTouchMode(false);

        textBin.clearFocus();
        textBin.setFocusable(true);
        textBin.setFocusableInTouchMode(true);
        textBin.requestFocus();

        int lenghtBin = textBin.getText().toString().length();
        if(lenghtBin == maxLengthforBin){
            mCurrentError = ErrorHandler.ErrorType.BIN_LENGTH;
            fullstack = true;
```

```
            limit = maxLengthforBin;
        }else if(lenghtBin > maxLengthforBin){
            clear();
        }else{
            mCurrentError = ErrorHandler.ErrorType.NULL;
            limit = lenghtBin;
            fullstack = false;
        }

        buttonDot.setEnabled(true);
        button0.setEnabled(true);
        button0.setTextColor(getResources().getColor(R.color.white));
        button1.setEnabled(true);
        button1.setTextColor(getResources().getColor(R.color.white));
        button2.setEnabled(false);
        button2.setTextColor(getResources().getColor(R.color.disable_cyan));
        button3.setEnabled(false);
        button3.setTextColor(getResources().getColor(R.color.disable_cyan));
        button4.setEnabled(false);
        button4.setTextColor(getResources().getColor(R.color.disable_cyan));
        button5.setEnabled(false);
        button5.setTextColor(getResources().getColor(R.color.disable_cyan));
        button6.setEnabled(false);
        button6.setTextColor(getResources().getColor(R.color.disable_cyan));
        button7.setEnabled(false);
        button7.setTextColor(getResources().getColor(R.color.disable_cyan));
        button8.setEnabled(false);
        button8.setTextColor(getResources().getColor(R.color.disable_cyan));
        button9.setEnabled(false);
        button9.setTextColor(getResources().getColor(R.color.disable_cyan));
        buttonA.setEnabled(false);
        buttonA.setTextColor(getResources().getColor(R.color.disable_light_blue));
        buttonB.setEnabled(false);
        buttonB.setTextColor(getResources().getColor(R.color.disable_light_blue));
        buttonC.setEnabled(false);
        buttonC.setTextColor(getResources().getColor(R.color.disable_light_blue));
        buttonD.setEnabled(false);
        buttonD.setTextColor(getResources().getColor(R.color.disable_light_blue));
        buttonE.setEnabled(false);
        buttonE.setTextColor(getResources().getColor(R.color.disable_light_blue));
        buttonF.setEnabled(false);
        buttonF.setTextColor(getResources().getColor(R.color.disable_light_blue));
    }
}

public class KeyboardClickListener implements View.OnClickListener{

    BaseState baseEnabled;
    public KeyboardClickListener(BaseState mBaseEnabled) {
```

```
        this.baseEnabled = mBaseEnabled;
    }

    public void onClick(View view)
    {
        switch (view.getId())
        {
            case R.id.zero:
                switch (baseEnabled)
                {
                    case DECIMAL:
                        cutter(mCurrentBase);
                        if(!fullstack){
                            textDec.setText(textDec.getText().toString() +
button0.getText().toString());
                        }else{
                            ErrorHandler.echoError(mCurrentError, activity);
                        }
                        break;
                    case HEXADECIMAL:
                        cutter(mCurrentBase);
                        if(!fullstack){
                            textHex.setText(textHex.getText().toString() +
button0.getText().toString());
                        }else{
                            ErrorHandler.echoError(mCurrentError, activity);
                        }
                        break;
                    case BINARY:
                        cutter(mCurrentBase);
                        if(!fullstack){
                            textBin.setText(textBin.getText().toString() + button0.getText().toString());
                        }else{
                            ErrorHandler.echoError(mCurrentError, activity);
                        }
                        break;
                }
                break;
            case R.id.one:
                switch (baseEnabled)
                {
                    case DECIMAL:
                        cutter(mCurrentBase);
                        if(!fullstack){
                            textDec.setText(textDec.getText().toString() +
button1.getText().toString());
                        }else{
                            ErrorHandler.echoError(mCurrentError, activity);
                        }
```

```
                    break;
                case HEXADECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textHex.setText(textHex.getText().toString() +
button1.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case BINARY:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textBin.setText(textBin.getText().toString() + button1.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
            }
            break;
        case R.id.two:
            switch (baseEnabled)
            {
                case DECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textDec.setText(textDec.getText().toString() +
button2.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case HEXADECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textHex.setText(textHex.getText().toString() +
button2.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case BINARY:
                    break;
            }
            break;
        case R.id.three:
            switch (baseEnabled)
            {
                case DECIMAL:
```

```
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textDec.setText(textDec.getText().toString() +
button3.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case HEXADECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textHex.setText(textHex.getText().toString() +
button3.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case BINARY:
                    break;
            }
            break;
        case R.id.four:
            switch (baseEnabled)
            {
                case DECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textDec.setText(textDec.getText().toString() +
button4.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case HEXADECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textHex.setText(textHex.getText().toString() +
button4.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case BINARY:
                    break;
            }
            break;
        case R.id.five:
            switch (baseEnabled)
            {
```

```
                case DECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textDec.setText(textDec.getText().toString() +
button5.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case HEXADECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textHex.setText(textHex.getText().toString() +
button5.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case BINARY:
                    break;
            }
            break;
        case R.id.six:
            switch (baseEnabled)
            {
                case DECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textDec.setText(textDec.getText().toString() +
button6.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case HEXADECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textHex.setText(textHex.getText().toString() +
button6.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case BINARY:
                    break;
            }
            break;
        case R.id.seven:
            switch (baseEnabled)
```

```
            {
               case DECIMAL:
                  cutter(mCurrentBase);
                  if(!fullstack){
                     textDec.setText(textDec.getText().toString() +
button7.getText().toString());
                  }else{
                     ErrorHandler.echoError(mCurrentError, activity);
                  }
                  break;
               case HEXADECIMAL:
                  cutter(mCurrentBase);
                  if(!fullstack){
                     textHex.setText(textHex.getText().toString() +
button7.getText().toString());
                  }else{
                     ErrorHandler.echoError(mCurrentError, activity);
                  }
                  break;
               case BINARY:
                  break;
            }
            break;
         case R.id.eight:
            switch (baseEnabled)
            {
               case DECIMAL:
                  cutter(mCurrentBase);
                  if(!fullstack){
                     textDec.setText(textDec.getText().toString() +
button8.getText().toString());
                  }else{
                     ErrorHandler.echoError(mCurrentError, activity);
                  }
                  break;
               case HEXADECIMAL:
                  cutter(mCurrentBase);
                  if(!fullstack){
                     textHex.setText(textHex.getText().toString() +
button8.getText().toString());
                  }else{
                     ErrorHandler.echoError(mCurrentError, activity);
                  }
                  break;
               case BINARY:
                  break;
            }
            break;
         case R.id.nine:
```

```
            switch (baseEnabled)
            {
               case DECIMAL:
                  cutter(mCurrentBase);
                  if(!fullstack){
                     textDec.setText(textDec.getText().toString() +
button9.getText().toString());
                  }else{
                     ErrorHandler.echoError(mCurrentError, activity);
                  }
                  break;
               case HEXADECIMAL:
                  cutter(mCurrentBase);
                  if(!fullstack){
                     textHex.setText(textHex.getText().toString() +
button9.getText().toString());
                  }else{
                     ErrorHandler.echoError(mCurrentError, activity);
                  }
                  break;
               case BINARY:
                  break;
            }
            break;
         case R.id.alpha:
            switch (baseEnabled)
            {
               case DECIMAL:
                  break;
               case HEXADECIMAL:
                  cutter(mCurrentBase);
                  if(!fullstack){
                     textHex.setText(textHex.getText().toString() +
buttonA.getText().toString());
                  }else{
                     ErrorHandler.echoError(mCurrentError, activity);
                  }
                  break;
               case BINARY:
                  break;
            }
            break;
         case R.id.beta:
            switch (baseEnabled)
            {
               case DECIMAL:
                  break;
               case HEXADECIMAL:
                  cutter(mCurrentBase);
```

```
                    if(!fullstack){
                        textHex.setText(textHex.getText().toString() +
buttonB.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case BINARY:
                    break;
            }
            break;
        case R.id.gamma:
            switch (baseEnabled)
            {
                case DECIMAL:
                    break;
                case HEXADECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textHex.setText(textHex.getText().toString() +
buttonC.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case BINARY:
                    break;
            }
            break;
        case R.id.delta:
            switch (baseEnabled)
            {
                case DECIMAL:
                    break;
                case HEXADECIMAL:
                    cutter(mCurrentBase);
                    if(!fullstack){
                        textHex.setText(textHex.getText().toString() +
buttonD.getText().toString());
                    }else{
                        ErrorHandler.echoError(mCurrentError, activity);
                    }
                    break;
                case BINARY:
                    break;
            }
            break;
        case R.id.epsilon:
            switch (baseEnabled)
```

```
            {
               case DECIMAL:
                  break;
               case HEXADECIMAL:
                  cutter(mCurrentBase);
                  if(!fullstack){
                     textHex.setText(textHex.getText().toString() +
buttonE.getText().toString());
                  }else{
                     ErrorHandler.echoError(mCurrentError, activity);
                  }
                  break;
               case BINARY:
                  break;
            }
            break;
         case R.id.fi:
            switch (baseEnabled)
            {
               case DECIMAL:
                  break;
               case HEXADECIMAL:
                  cutter(mCurrentBase);
                  if(!fullstack){
                     textHex.setText(textHex.getText().toString() +
buttonF.getText().toString());
                  }else{
                     ErrorHandler.echoError(mCurrentError, activity);
                  }
                  break;
               case BINARY:
                  break;
            }
            break;
         case R.id.dot:
            switch (baseEnabled)
            {
               case DECIMAL:
                  textDec.setText(textDec.getText().toString() +
buttonDot.getText().toString());
                  break;
               case HEXADECIMAL:
                  textHex.setText(textHex.getText().toString() +
buttonDot.getText().toString());
                  break;
               case BINARY:
                  textBin.setText(textBin.getText().toString() + buttonDot.getText().toString());
                  break;
            }
```

```
            break;
         case R.id.delete:
            switch (baseEnabled)
            {
               case DECIMAL:
                  delete(textDec);
                  break;
               case HEXADECIMAL:
                  delete(textHex);
                  break;
               case BINARY:
                  delete(textBin);
                  break;
            }
            break;
         case R.id.clear:
            clear();
      }
   }
}

protected BaseState getBase(){return mCurrentBase;}

protected void setBase(BaseState state) {
   if (mCurrentBase != state) {
      mCurrentBase = state;
   }
}

protected ConverterState getState(){return mCurrentState;}

protected void setState(ConverterState state) {
   BaseState mBase;
   if (mCurrentState != state) {
      mCurrentState = state;
      if(mCurrentState.equals(ConverterState.READY)){
         mBase = getBase();
         switch (mBase){
            case DECIMAL:
               if(!textDec.hasFocus()){
                  textAscii.clearFocus();
                  textBin.clearFocus();
                  textHex.clearFocus();
                  textDec.clearFocus();
                  textDec.setFocusable(true);
                  textDec.setFocusableInTouchMode(true);
                  textDec.requestFocus();
               }
               break;
```

```
      case HEXADECIMAL:
        if(!textHex.hasFocus()){
          textAscii.clearFocus();
          textBin.clearFocus();
          textDec.clearFocus();
          textHex.clearFocus();
          textHex.setFocusable(true);
          textHex.setFocusableInTouchMode(true);
          textHex.requestFocus();
        }
        break;
      case BINARY:
        if(!textBin.hasFocus()){
          textAscii.clearFocus();
          textHex.clearFocus();
          textDec.clearFocus();
          textBin.clearFocus();
          textBin.setFocusable(true);
          textBin.setFocusableInTouchMode(true);
          textBin.requestFocus();
        }
        break;
    }
}
else if(mCurrentState.equals(ConverterState.EVALUATE)){}
else if(mCurrentState.equals(ConverterState.ERROR)){
    mBase = getBase();
    switch (mBase){
      case DECIMAL:
        textDec.clearFocus();
        button0.setOnClickListener(this);
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
        button3.setOnClickListener(this);
        button4.setOnClickListener(this);
        button5.setOnClickListener(this);
        button6.setOnClickListener(this);
        button7.setOnClickListener(this);
        button8.setOnClickListener(this);
        button9.setOnClickListener(this);
        buttonA.setOnClickListener(null);
        buttonB.setOnClickListener(null);
        buttonC.setOnClickListener(null);
        buttonD.setOnClickListener(null);
        buttonE.setOnClickListener(null);
        buttonF.setOnClickListener(null);
        buttonClear.setOnClickListener(this);
        buttonDelete.setOnClickListener(this);
        buttonDot.setOnClickListener(null);
```

```
            textDec.setText(getResources().getString(R.string.invalid_input));
            break;
        case HEXADECIMAL:
            textHex.clearFocus();
            button0.setOnClickListener(this);
            button1.setOnClickListener(this);
            button2.setOnClickListener(this);
            button3.setOnClickListener(this);
            button4.setOnClickListener(this);
            button5.setOnClickListener(this);
            button6.setOnClickListener(this);
            button7.setOnClickListener(this);
            button8.setOnClickListener(this);
            button9.setOnClickListener(this);
            buttonA.setOnClickListener(this);
            buttonB.setOnClickListener(this);
            buttonC.setOnClickListener(this);
            buttonD.setOnClickListener(this);
            buttonE.setOnClickListener(this);
            buttonF.setOnClickListener(this);
            buttonClear.setOnClickListener(this);
            buttonDelete.setOnClickListener(this);
            buttonDot.setOnClickListener(null);
            textHex.setText(getResources().getString(R.string.invalid_input));
            break;
        case BINARY:
            textBin.clearFocus();
            button0.setOnClickListener(this);
            button1.setOnClickListener(this);
            button2.setOnClickListener(null);
            button3.setOnClickListener(null);
            button4.setOnClickListener(null);
            button5.setOnClickListener(null);
            button6.setOnClickListener(null);
            button7.setOnClickListener(null);
            button8.setOnClickListener(null);
            button9.setOnClickListener(null);
            buttonA.setOnClickListener(null);
            buttonB.setOnClickListener(null);
            buttonC.setOnClickListener(null);
            buttonD.setOnClickListener(null);
            buttonE.setOnClickListener(null);
            buttonF.setOnClickListener(null);
            buttonClear.setOnClickListener(this);
            buttonDelete.setOnClickListener(this);
            buttonDot.setOnClickListener(null);
            textBin.setText(getResources().getString(R.string.invalid_input));
            break;
    }
```

```
        }
    }
}

@Override
public void onClick(View v) {
    BaseState mBase = getBase();
    switch (v.getId()) {
        case R.id.clear:
            clear();
            switch (mBase){
                case DECIMAL:
                    if(mCurrentState.equals(ConverterState.ERROR)){
                        setState(ConverterState.EVALUATE);
                    }
                    textDec.setFocusableInTouchMode(true);
                    textDec.requestFocus();
                    break;
                case HEXADECIMAL:
                    if(mCurrentState.equals(ConverterState.ERROR)){
                        setState(ConverterState.EVALUATE);
                    }
                    textHex.setFocusableInTouchMode(true);
                    textHex.requestFocus();
                    break;
                case BINARY:
                    if(mCurrentState.equals(ConverterState.ERROR)){
                        setState(ConverterState.EVALUATE);
                    }
                    textBin.setFocusableInTouchMode(true);
                    textBin.requestFocus();
                    break;
            }
            break;
        case R.id.delete:
            clear();
            switch (mBase){
                case DECIMAL:
                    if(mCurrentState.equals(ConverterState.ERROR)){
                        setState(ConverterState.EVALUATE);
                    }
                    textDec.setFocusableInTouchMode(true);
                    textDec.requestFocus();
                    break;
                case HEXADECIMAL:
                    if(mCurrentState.equals(ConverterState.ERROR)){
                        setState(ConverterState.EVALUATE);
                    }
                    textHex.setFocusableInTouchMode(true);
```

```
                textHex.requestFocus();
                break;
            case BINARY:
                if(mCurrentState.equals(ConverterState.ERROR)){
                    setState(ConverterState.EVALUATE);
                }
                textBin.setFocusableInTouchMode(true);
                textBin.requestFocus();
                break;
        }
        break;
    default:
        switch (mBase){
            case DECIMAL:
                if(mCurrentState.equals(ConverterState.ERROR)){
                    setState(ConverterState.EVALUATE);
                }
                clear();
                textDec.setFocusableInTouchMode(true);
                textDec.requestFocus();
                textDec.setText(((Button) v).getText().toString());
                break;
            case HEXADECIMAL:
                if(mCurrentState.equals(ConverterState.ERROR)){
                    setState(ConverterState.EVALUATE);
                }
                clear();
                textHex.setFocusableInTouchMode(true);
                textHex.requestFocus();
                textHex.setText(((Button) v).getText().toString());
                break;
            case BINARY:
                if(mCurrentState.equals(ConverterState.ERROR)){
                    setState(ConverterState.EVALUATE);
                }
                clear();
                textBin.setFocusableInTouchMode(true);
                textBin.requestFocus();
                textBin.setText(((Button) v).getText().toString());
                break;
        }
        break;
    }
}

protected void delete(EditText currentEditText){
    String editTextlength = currentEditText.getText().toString();
    try {
        if (editTextlength.length() > 0) {
```

```
            currentEditText.setText(editTextlength.substring(0, editTextlength.length() - 1));
            currentEditText.setSelection(currentEditText.getText().length());
            switch(mCurrentBase){
               case DECIMAL:
                  if(limit >= maxLengthforDec){
                     limit--;
                     if(limit < maxLengthforDec){
                        fullstack = false;
                     }
                  }else{
                     limit--;
                     fullstack = false;
                  }
                  break;
               case HEXADECIMAL:
                  if(limit >= maxLengthforHex){
                     limit--;
                     if(limit < maxLengthforHex){
                        fullstack = false;
                     }
                  }else{
                     limit--;
                     fullstack = false;
                  }
                  break;
               case BINARY:
                  if(limit >= maxLengthforBin){
                     limit--;
                     if(limit < maxLengthforBin){
                        fullstack = false;
                     }
                  }else{
                     limit--;
                     fullstack = false;
                  }
                  break;
            }
         }
      } catch (Exception e) {
         System.out.print(e);
      }
   }

   protected void clear(){
      textDec.setText(null);
      textBin.setText(null);
      textHex.setText(null);
      textAscii.setText(null);
      mCurrentError = ErrorHandler.ErrorType.NULL;
```

```
    fullstack = false;
    limit = 0;
}

private void cutter(BaseState base){
    if(!fullstack){
        limit++;
    }
    switch (base){
        case DECIMAL:
            if((limit > maxLengthforDec) && !fullstack){
                mCurrentError = ErrorHandler.ErrorType.DEC_LENGTH;
                fullstack = true;
                limit = maxLengthforDec;
            }
            break;
        case HEXADECIMAL:
            if((limit > maxLengthforHex) && !fullstack){
                mCurrentError = ErrorHandler.ErrorType.HEX_LENGTH;
                fullstack = true;
                limit = maxLengthforHex;
            }
            break;
        case BINARY:
            if((limit > maxLengthforBin) && !fullstack){
                mCurrentError = ErrorHandler.ErrorType.BIN_LENGTH;
                fullstack = true;
                limit = maxLengthforBin;
            }
            break;
    }
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

private enum BaseState {
```

```
        DECIMAL, HEXADECIMAL, BINARY
    }

    private enum ConverterState{
        READY, EVALUATE, ERROR
    }

}
```

MAINACTIVITY.JAVA

```
package com.calcobin;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.view.WindowManager;

import com.google.firebase.analytics.FirebaseAnalytics;

public class MainActivity extends FragmentActivity {

   public static ViewPager pager;
   private FirebaseAnalytics mFirebaseAnalytics;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      //disable keyboard
      getWindow().setFlags(WindowManager.LayoutParams.FLAG_ALT_FOCUSABLE_IM,
WindowManager.LayoutParams.FLAG_ALT_FOCUSABLE_IM);

      mFirebaseAnalytics = FirebaseAnalytics.getInstance(this);

      pager = (ViewPager) findViewById(R.id.viewPager);
      pager.setAdapter(new MyPagerAdapter(getSupportFragmentManager()));

      if(pager.getCurrentItem() != 1){
         pager.setCurrentItem(1);
      }
   }

   @Override
   public void onBackPressed() {
      if(MainActivity.pager.getCurrentItem() == 0 || MainActivity.pager.getCurrentItem() ==
2) {
         MainActivity.pager.setCurrentItem(1, true);
      } else {
         super.onBackPressed(); // This will pop the Activity from the stack.
      }
   }
```

```
    private class MyPagerAdapter extends FragmentPagerAdapter {

        public MyPagerAdapter(FragmentManager fm) {
            super(fm);
        }

        @Override
        public Fragment getItem(int pos) {
            switch(pos) {
                case 0: return new Converter();
                case 1: return new Calculator();
                case 2: return new Result();
                default: return new Calculator();
            }
        }

        @Override
        public int getCount() {
            return 3;
        }
    }
}
```

RESULT.JAVA

```
package com.calcobin;

import android.content.ClipData;
import android.graphics.Canvas;
import android.graphics.Color;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.text.InputFilter;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;

public class Result extends Fragment {

   static EditText editText_dec_result, editText_hex_result, editText_bin_result;


   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
      View v = inflater.inflate(R.layout.activity_result, container, false);

      editText_dec_result = (EditText) v.findViewById(R.id.decimalResult_editText);
      editText_hex_result = (EditText) v.findViewById(R.id.hexResult_editText);
      editText_bin_result = (EditText) v.findViewById(R.id.binResult_editText);

      editText_hex_result.setFilters(new InputFilter[] {new InputFilter.AllCaps()});

      return v;
   }

   private final class MyTouchListener implements View.OnTouchListener {
      public boolean onTouch(View view, MotionEvent motionEvent) {
         if ((motionEvent.getAction() == MotionEvent.ACTION_DOWN) ||
(motionEvent.getAction() == MotionEvent.ACTION_UP) || (motionEvent.getAction() ==
MotionEvent.ACTION_MOVE) || (motionEvent.getAction() ==
MotionEvent.ACTION_CANCEL)) {
            ClipData data = ClipData.newPlainText("", "");
            View.DragShadowBuilder shadowBuilder = new View.DragShadowBuilder(view);
            Canvas mCanvas = new Canvas();
            mCanvas.drawColor(Color.TRANSPARENT);
            shadowBuilder.onDrawShadow(mCanvas);
            view.startDrag(data, shadowBuilder, view, 0);
            return true;
```

```
            } else {
                ClipData data = ClipData.newPlainText("", "");
                View.DragShadowBuilder shadowBuilder = new View.DragShadowBuilder(view);
                Canvas mCanvas = new Canvas();
                mCanvas.drawColor(Color.TRANSPARENT);
                shadowBuilder.onDrawShadow(mCanvas);
                view.startDrag(data, shadowBuilder, view, 0);
                return true;
            }
        }
    }


}
```

BLUE_BUTTON.XML

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
   <item>
      <shape android:shape="rectangle">
         <solid android:color="@color/border" />
      </shape>
   </item>
   <item android:right="0px" android:bottom="0px">
      <shape android:shape="rectangle">
         <solid android:color="@color/blue" />
      </shape>
   </item>
</layer-list>
```

dark_blue_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
<item>
   <shape android:shape="rectangle">
      <solid android:color="@color/border" />
   </shape>
</item>
<item android:right="0px" android:bottom="0px">
   <shape android:shape="rectangle">
      <solid android:color="@color/dark_blue" />
   </shape>
</item>
</layer-list>
```

DECIMAL_KEYBOARD_BUTTON.XML

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
   <item>
      <shape android:shape="rectangle">
         <solid android:color="@color/border" />
      </shape>
   </item>
   <item android:right="0px" android:bottom="0px">
      <shape android:shape="rectangle">
         <solid android:color="@color/cyan" />
      </shape>
   </item>
</layer-list>
```

edittextresults.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
   <item>
      <shape android:shape="rectangle">
         <solid android:color="@color/border" />
      </shape>
   </item>
   <item android:bottom="0px">
      <shape android:shape="rectangle">
         <solid android:color="@color/white" />
      </shape>
   </item>
</layer-list>
```

LETTER_KEYBOARD_BUTTON.XML

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
   <item>
      <shape android:shape="rectangle">
         <solid android:color="@color/border" />
      </shape>
   </item>
   <item android:right="0px" android:bottom="0px">
      <shape android:shape="rectangle">
         <solid android:color="@color/light_blue" />
      </shape>
   </item>
</layer-list>
```

mustard_keyboard_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
   <item>
      <shape android:shape="rectangle">
         <solid android:color="@color/border" />
      </shape>
   </item>
   <item android:right="0px" android:bottom="0px">
      <shape android:shape="rectangle">
         <solid android:color="@color/mustard" />
      </shape>
   </item>
</layer-list>
```

ACTIVITY_CALCULATOR.XML

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="com.calcobin.Calculator"
   android:orientation="vertical"
   android:id="@+id/linear_Calculator_Layout"
   android:weightSum="12">

   <LinearLayout
      android:orientation="vertical"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/displayScreenLay"
      android:background="@drawable/edittextresults"
      android:layout_weight="9.5"
      android:weightSum="6">

      <EditText
         android:text="0"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:id="@+id/editTextExpession"
         android:gravity="right|center_vertical"
         android:layout_gravity="right"
         android:textSize="20sp"
         android:enabled="false"
         android:focusableInTouchMode="false"
         android:focusable="false"
         android:cursorVisible="false"
         android:background="@android:color/transparent"
         android:layout_weight="4"
         android:paddingRight="10sp"
         android:singleLine="true"
         android:scrollHorizontally="true"
         android:inputType="text" />

      <EditText
         android:text="@string/zero"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:id="@+id/editTextInput"
         android:gravity="right|center_vertical"
         android:layout_gravity="right"
         android:textSize="33sp"
         android:cursorVisible="false"
```

```
        android:background="@android:color/transparent"
        android:layout_weight="2"
        android:paddingRight="10sp"
        android:singleLine="true"
        android:scrollHorizontally="true" />
</LinearLayout>

<LinearLayout
    android:id="@+id/calculator_keyboard"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="2.5"
    android:orientation="vertical">

    <LinearLayout
        android:id="@+id/firstRow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1">

        <Button
            android:id="@+id/decimalNum"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@drawable/letter_keyboard_button"
            android:fontFamily="sans-serif-light"
            android:text="@string/decimal"
            android:textColor="@color/white"
            android:textSize="24sp" />

        <Button
            android:id="@+id/hexNum"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@drawable/letter_keyboard_button"
            android:fontFamily="sans-serif-light"
            android:text="@string/hexadecimal"
            android:textColor="@color/white"
            android:textSize="22sp" />

        <Button
            android:id="@+id/binaryNum"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@drawable/letter_keyboard_button"
            android:fontFamily="sans-serif-light"
```

```
        android:text="@string/binary"
        android:textColor="@color/white"
        android:textSize="22sp" />
    </LinearLayout>

    <LinearLayout
      android:id="@+id/eraseRow"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_weight="1">

      <Button
        android:id="@+id/ce"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/blue_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/ce"
        android:textColor="@color/white"
        android:textSize="24sp" />

      <Button
        android:id="@+id/clear"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/blue_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/clear"
        android:textColor="@color/white"
        android:textSize="24sp" />

      <Button
        android:id="@+id/backspace"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/blue_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/delete"
        android:textColor="@color/white"
        android:textSize="24sp" />

    </LinearLayout>

    <LinearLayout
      android:id="@+id/secondRow"
      android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:layout_weight="1">


<Button
   android:id="@+id/plus"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_weight="1"
   android:background="@drawable/dark_blue_button"
   android:fontFamily="sans-serif-light"
   android:text="@string/plus"
   android:textColor="@color/white"
   android:textSize="27sp" />

<Button
   android:id="@+id/minus"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_weight="1"
   android:background="@drawable/dark_blue_button"
   android:fontFamily="sans-serif-light"
   android:text="@string/minus"
   android:textColor="@color/white"
   android:textSize="27sp" />

<Button
   android:id="@+id/multiply"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_weight="1"
   android:background="@drawable/dark_blue_button"
   android:fontFamily="sans-serif-light"
   android:text="@string/multiply"
   android:textAllCaps="false"
   android:textColor="@color/white"
   android:textSize="27sp" />

<Button
   android:id="@+id/divide"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_weight="1"
   android:background="@drawable/dark_blue_button"
   android:fontFamily="sans-serif-light"
   android:text="@string/divide"
   android:textColor="@color/white"
   android:textSize="27sp" />
```

```
</LinearLayout>

<LinearLayout
   android:id="@+id/thirdRow"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_weight="1">

   <Button
      android:id="@+id/seven"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="1"
      android:background="@drawable/decimal_keyboard_button"
      android:fontFamily="sans-serif-light"
      android:text="@string/seven"
      android:textColor="@color/white"
      android:textSize="27sp" />

   <Button
      android:id="@+id/eigth"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="1"
      android:background="@drawable/decimal_keyboard_button"
      android:fontFamily="sans-serif-light"
      android:text="@string/eight"
      android:textColor="@color/white"
      android:textSize="27sp" />

   <Button
      android:id="@+id/nine"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="1"
      android:background="@drawable/decimal_keyboard_button"
      android:fontFamily="sans-serif-light"
      android:text="@string/nine"
      android:textColor="@color/white"
      android:textSize="27sp" />

   <Button
      android:id="@+id/alpha"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="1"
      android:background="@drawable/letter_keyboard_button"
      android:fontFamily="sans-serif-light"
      android:text="@string/a"
```

```
        android:textColor="@color/white"
        android:textSize="27sp" />

    <Button
        android:id="@+id/delta"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/letter_keyboard_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/d"
        android:textColor="@color/white"
        android:textSize="27sp" />
</LinearLayout>

<LinearLayout
    android:id="@+id/forthRow"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1">

    <Button
        android:id="@+id/four"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/decimal_keyboard_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/four"
        android:textColor="@color/white"
        android:textSize="27sp" />

    <Button
        android:id="@+id/five"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/decimal_keyboard_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/five"
        android:textColor="@color/white"
        android:textSize="27sp" />

    <Button
        android:id="@+id/six"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/decimal_keyboard_button"
```

```
        android:fontFamily="sans-serif-light"
        android:text="@string/six"
        android:textColor="@color/white"
        android:textSize="27sp" />

    <Button
        android:id="@+id/beta"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/letter_keyboard_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/b"
        android:textColor="@color/white"
        android:textSize="27sp" />

    <Button
        android:id="@+id/epsilon"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/letter_keyboard_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/e"
        android:textColor="@color/white"
        android:textSize="27sp" />
</LinearLayout>

<LinearLayout
    android:id="@+id/fifthRow"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1">

    <Button
        android:id="@+id/one"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/decimal_keyboard_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/one"
        android:textColor="@color/white"
        android:textSize="27sp" />

    <Button
        android:id="@+id/two"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        android:layout_weight="1"
        android:background="@drawable/decimal_keyboard_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/two"
        android:textColor="@color/white"
        android:textSize="27sp" />

    <Button
        android:id="@+id/three"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/decimal_keyboard_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/three"
        android:textColor="@color/white"
        android:textSize="27sp" />

    <Button
        android:id="@+id/gamma"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/letter_keyboard_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/c"
        android:textColor="@color/white"
        android:textSize="27sp" />

    <Button
        android:id="@+id/fi"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/letter_keyboard_button"
        android:fontFamily="sans-serif-light"
        android:text="@string/f"
        android:textColor="@color/white"
        android:textSize="27sp" />
</LinearLayout>

<LinearLayout
    android:id="@+id/sixthRow"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1">

    <Button
        android:id="@+id/zero"
```

```
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1.33399"
            android:background="@drawable/decimal_keyboard_button"
            android:fontFamily="sans-serif-light"
            android:text="@string/zero"
            android:textColor="@color/white"
            android:textSize="27sp" />

        <Button
            android:id="@+id/dot"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1.33399"
            android:background="@drawable/decimal_keyboard_button"
            android:fontFamily="sans-serif-light"
            android:text="@string/dot"
            android:textColor="@color/white"
            android:textSize="27sp" />

        <Button
            android:id="@+id/plus_minus"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1.33399"
            android:background="@drawable/blue_button"
            android:fontFamily="sans-serif-light"
            android:text="@string/plus_minus"
            android:textColor="@color/white"
            android:textSize="27sp" />

        <Button
            android:id="@+id/equal"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@drawable/mustard_keyboard_button"
            android:fontFamily="sans-serif-light"
            android:text="@string/equal"
            android:textColor="@color/white"
            android:textSize="27sp" />

    </LinearLayout>
  </LinearLayout>
</LinearLayout>
```

ACTIVITY_CONVERTER.XML

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context="com.calcobin.Converter"
   android:weightSum="12"
   android:id="@+id/linear_Converter_Layout">

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/results"
      android:orientation="vertical"
      android:layout_weight="5.43"
      android:background="@color/white">


      <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:id="@+id/layoutDec"
         android:layout_weight="1">

         <Button
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="@string/decimal"
            android:id="@+id/buttonDec"
            android:layout_weight="2"
            android:background="@drawable/dark_blue_button"
            android:textColor="@color/white"
            android:fontFamily="sans-serif-condensed"
            android:textSize="24sp" />

         <EditText
            android:id="@+id/editTextDec"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@drawable/edittextresults"
            android:cursorVisible="true"
            android:fontFamily="sans-serif"
            android:gravity="left|center"
            android:inputType="text|textFilter"
```

```
        android:maxLines="1"
        android:padding="5dp"
        android:scrollHorizontally="true"
        android:text="01234567890123456789012345678"
        android:textIsSelectable="false"
        android:textSize="18sp" />

    </LinearLayout>

    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/layoutHex"
      android:layout_weight="1">

      <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="2"
        android:text="@string/hexadecimal"
        android:id="@+id/buttonHex"
        android:background="@drawable/dark_blue_button"
        android:elevation="1dp"
        android:textColor="@color/white"
        android:fontFamily="sans-serif-condensed"
        android:textSize="24sp" />

      <EditText
        android:id="@+id/editTextHex"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/edittextresults"
        android:clickable="false"
        android:fontFamily="sans-serif"
        android:gravity="left|center"
        android:inputType="text|textFilter"
        android:maxLines="1"
        android:padding="5dp"
        android:scrollHorizontally="true"
        android:text="012345678910"
        android:textIsSelectable="false"
        android:textSize="18sp" />
    </LinearLayout>

    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/layoutBin"
```

```
    android:layout_weight="1">

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/binary"
        android:layout_weight="2"
        android:id="@+id/buttonBin"
        android:background="@drawable/dark_blue_button"
        android:textColor="@color/white"
        android:fontFamily="sans-serif-condensed"
        android:textSize="24sp" />

    <EditText
        android:id="@+id/editTextBin"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@drawable/edittextresults"
        android:clickable="false"
        android:fontFamily="sans-serif"
        android:gravity="left|center"
        android:inputType="text|textFilter"
        android:maxLines="1"
        android:padding="5dp"
        android:text="01234567891011121314"
        android:textIsSelectable="false"
        android:textSize="18sp" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/layoutAscii"
    android:layout_weight="1">

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/ascii"
        android:layout_weight="2"
        android:id="@+id/buttonAscii"
        android:background="@drawable/dark_blue_button"
        android:textColor="@color/white"
        android:fontFamily="sans-serif-condensed"
        android:textSize="24sp" />

    <EditText
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:id="@+id/editTextAscii"
        android:textIsSelectable="false"
        android:padding="5dp"
        android:textSize="18sp"
        android:background="@drawable/edittextresults"
        android:fontFamily="sans-serif"
        android:maxLines="1"
        android:clickable="false"
        android:gravity="left|center"
        android:inputType="text" />
    </LinearLayout>

</LinearLayout>


<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/keyboard"
    android:orientation="vertical"
    android:layout_weight="6.57">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/frow"
        android:layout_weight="1">

        <Button
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="@string/seven"
            android:id="@+id/seven"
            android:layout_weight="1"
            android:fontFamily="sans-serif-light"
            android:background="@drawable/decimal_keyboard_button"
            android:textColor="@color/white"
            android:textSize="27sp" />

        <Button
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="@string/eight"
            android:id="@+id/eight"
            android:layout_weight="1"
            android:fontFamily="sans-serif-light"
            android:textSize="27sp"
```

```
        android:background="@drawable/decimal_keyboard_button"
        android:textColor="@color/white" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/nine"
        android:id="@+id/nine"
        android:layout_weight="1"
        android:fontFamily="sans-serif-light"
        android:background="@drawable/decimal_keyboard_button"
        android:textColor="@color/white"
        android:textSize="27sp" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/a"
        android:id="@+id/alpha"
        android:layout_weight="1"
        android:fontFamily="sans-serif-light"
        android:textSize="27sp"
        android:background="@drawable/letter_keyboard_button"
        android:textColor="@color/white" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/d"
        android:id="@+id/delta"
        android:layout_weight="1"
        android:fontFamily="sans-serif-light"
        android:textSize="27sp"
        android:background="@drawable/letter_keyboard_button"
        android:textColor="@color/white" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/srow"
    android:layout_weight="1">

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/four"
        android:id="@+id/four"
        android:layout_weight="1"
```

```
        android:fontFamily="sans-serif-light"
        android:textSize="27sp"
        android:background="@drawable/decimal_keyboard_button"
        android:textColor="@color/white" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/five"
        android:id="@+id/five"
        android:layout_weight="1"
        android:fontFamily="sans-serif-light"
        android:textSize="27sp"
        android:background="@drawable/decimal_keyboard_button"
        android:textColor="@color/white" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/six"
        android:id="@+id/six"
        android:layout_weight="1"
        android:fontFamily="sans-serif-light"
        android:textSize="27sp"
        android:background="@drawable/decimal_keyboard_button"
        android:textColor="@color/white" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/b"
        android:id="@+id/beta"
        android:layout_weight="1"
        android:fontFamily="sans-serif-light"
        android:textSize="27sp"
        android:background="@drawable/letter_keyboard_button"
        android:textColor="@color/white" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/e"
        android:id="@+id/epsilon"
        android:layout_weight="1"
        android:fontFamily="sans-serif-light"
        android:textSize="27sp"
        android:background="@drawable/letter_keyboard_button"
        android:textColor="@color/white" />
</LinearLayout>
```

```
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:id="@+id/trow"
   android:layout_weight="1">

   <Button
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:text="@string/one"
      android:id="@+id/one"
      android:layout_weight="1"
      android:fontFamily="sans-serif-light"
      android:textSize="27sp"
      android:background="@drawable/decimal_keyboard_button"
      android:textColor="@color/white" />

   <Button
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:text="@string/two"
      android:id="@+id/two"
      android:layout_weight="1"
      android:fontFamily="sans-serif-light"
      android:textSize="27sp"
      android:background="@drawable/decimal_keyboard_button"
      android:textColor="@color/white" />

   <Button
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:text="@string/three"
      android:id="@+id/three"
      android:layout_weight="1"
      android:fontFamily="sans-serif-light"
      android:textSize="27sp"
      android:background="@drawable/decimal_keyboard_button"
      android:textColor="@color/white" />

   <Button
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:text="@string/c"
      android:id="@+id/gamma"
      android:layout_weight="1"
      android:fontFamily="sans-serif-light"
      android:textSize="27sp"
      android:background="@drawable/letter_keyboard_button"
```

```
        android:textColor="@color/white" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/f"
        android:id="@+id/fi"
        android:layout_weight="1"
        android:fontFamily="sans-serif-light"
        android:textSize="27sp"
        android:background="@drawable/letter_keyboard_button"
        android:textColor="@color/white" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/ffrow"
    android:layout_weight="1">

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/zero"
        android:id="@+id/zero"
        android:layout_weight="1.33399"
        android:fontFamily="sans-serif-light"
        android:textSize="27sp"
        android:background="@drawable/decimal_keyboard_button"
        android:textColor="@color/white" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/dot"
        android:id="@+id/dot"
        android:layout_weight="1.33399"
        android:fontFamily="sans-serif-light"
        android:textSize="27sp"
        android:background="@drawable/decimal_keyboard_button"
        android:textColor="@color/white" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/clear"
        android:id="@+id/clear"
        android:layout_weight="1.33399"
        android:fontFamily="sans-serif-light"
```

```
            android:textSize="27sp"
            android:background="@drawable/blue_button"
            android:textColor="@color/white" />

        <Button
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="@string/delete"
            android:id="@+id/delete"
            android:layout_weight="1"
            android:fontFamily="sans-serif-light"
            android:textSize="27sp"
            android:background="@drawable/mustard_keyboard_button"
            android:textColor="@color/white" />
      </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

ACTIVITY_MAIN.XML

```
<android.support.v4.view.ViewPager
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:id="@+id/viewPager"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  />
```

FRAGMENT_MAIN_PAGE.XML

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="com.calcobin.MainPageFragment">
</LinearLayout>
```

ACTIVITY_RESULT.XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/linear_Result_Layout"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context="com.calcobin.Result"
   android:weightSum="12">

   <LinearLayout
      android:orientation="vertical"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:background="@color/cyan"
      android:layout_weight="3">

      <TextView
         android:text="@string/result"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:id="@+id/resultTitle_textView"
         android:padding="20sp"
         android:fontFamily="sans-serif-condensed"
         android:gravity="center"
         android:enabled="true"
         android:cursorVisible="false"
         android:textColor="@color/white"
         android:textSize="40sp" />
   </LinearLayout>

   <LinearLayout
      android:orientation="vertical"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:background="@color/light_blue"
      android:layout_weight="3"
      android:weightSum="12">

      <TextView
         android:text="@string/decimal_base"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:id="@+id/decimalResult_textView"
         android:gravity="center"
         android:padding="10sp"
```

```
        android:textColor="@color/white"
        android:textSize="18sp"
        android:layout_weight="6" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/decimalResult_editText"
        android:gravity="center"
        android:cursorVisible="false"
        android:textColor="@color/white"
        android:textIsSelectable="true"
        android:clickable="false"
        android:textSize="14sp"
        android:padding="10sp"
        android:layout_weight="6"
        android:background="@android:color/transparent" />


</LinearLayout>

<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/blue"
    android:layout_weight="3"
    android:weightSum="12">

    <TextView
        android:text="@string/hexadecimal_base"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/hexResult_textView"
        android:gravity="center"
        android:padding="10sp"
        android:textColor="@color/white"
        android:textSize="18sp"
        android:layout_weight="6" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:ems="10"
        android:id="@+id/hexResult_editText"
        android:gravity="center"
        android:cursorVisible="false"
        android:textColor="@color/white"
        android:textSize="14sp"
```

```
        android:padding="10sp"
        android:layout_weight="6"
        android:background="@android:color/transparent" />
    </LinearLayout>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/dark_blue"
        android:layout_weight="3"
        android:weightSum="12">

        <TextView
            android:text="@string/binary_base"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/binResult_textView"
            android:gravity="center"
            android:padding="10sp"
            android:textColor="@color/white"
            android:textSize="18sp"
            android:layout_weight="6" />

        <EditText
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/binResult_editText"
            android:gravity="center"
            android:cursorVisible="false"
            android:textColor="@color/white"
            android:textSize="14sp"
            android:padding="10sp"
            android:layout_weight="6"
            android:background="@android:color/transparent" />
    </LinearLayout>
</LinearLayout>
```

COLORS.XML

```
<resources>

    <color name="black_overlay">#66000000</color>
    <color name="white">#ffffff</color>
    <color name="black">#000000</color>
    <color name="border">#aaaaaa</color>
    <color name="green">#00ffae</color>
    <color name="gray">#e1e1e1</color>
    <color name="disable_blue">#c6cfd7</color>
    <color name="disable_gray">#c5c5c5</color>
    <color name="disable_cyan">#a3ecdf</color>
    <color name="disable_light_blue">#80d0ec</color>
    <color name="light_gray">#aaaaaa</color>

    <!--pallete-->
    <color name="blue">#05518b</color>
    <color name="light_blue">#00a1d8</color>
    <color name="dark_blue">#003056</color>
    <color name="cyan">#46d9bf</color>
    <color name="mustard">#f2d03b</color>


</resources>
```

STRINGS.XML

```
<resources>
    <string name="app_name">CalCoBin</string>

    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="calculator_menu">Calculator</string>
    <string name="converter_menu">Converter</string>
    <string name="learn_menu">Learn More</string>
    <string name="about_menu">About</string>
    <string name="result">Result</string>

    <string name="title_activity_fullscreen">FullscreenActivity</string>
    <string name="dummy_button">Dummy Button</string>
    <string name="dummy_content">DUMMY\nCONTENT</string>
    <string name="title_activity_splash">Calcobin</string>
    <string name="title_activity_calculator">Calculator</string>
    <string name="title_activity_converter">Converter</string>
    <string name="title_activity_result">Result</string>

    <!--Base-->
    <string name="decimal_base">Decimal</string>
    <string name="hexadecimal_base">Hexadecimal</string>
    <string name="binary_base">Binary</string>

    <!--Calculator/Converter-->
    <string name="one">1</string>
    <string name="two">2</string>
    <string name="three">3</string>
    <string name="four">4</string>
    <string name="five">5</string>
    <string name="six">6</string>
    <string name="seven">7</string>
    <string name="eight">8</string>
    <string name="nine">9</string>
    <string name="zero">0</string>
    <string name="a">A</string>
    <string name="b">B</string>
    <string name="c">C</string>
    <string name="d">D</string>
    <string name="e">E</string>
    <string name="f">F</string>
    <string name="ce">CE</string>
    <string name="clear">C</string>
    <string name="backspace">&lt;--</string>
    <string name="plus_minus">+/-</string>
    <string name="divide">÷</string>
```

```
    <string name="multiply">×</string>
    <string name="minus">–</string>
    <string name="plus">+</string>
    <string name="dot">.</string>
    <string name="equal">=</string>
    <string name="erase">&amp;lt;&amp;#8212;</string>
    <string name="delete">DEL</string>
    <string name="decimal">DEC</string>
    <string name="binary">BIN</string>
    <string name="hexadecimal">HEX</string>
    <string name="ascii">ASCII</string>

    <!-- Messages -->
    <string name="hello_blank_fragment">Hello blank fragment</string>
    <string name="invalid_input">Invalid Input</string>
    <string name="ascii_message">You cannot input ASCII code&amp;#8230; yet!</string>
    <string name="dividebyzero">You cannot divide by zero</string>
    <string name="overlay">Overlay</string>
    <string name="overlay_icon">Overlay icon</string>
</resources>
```