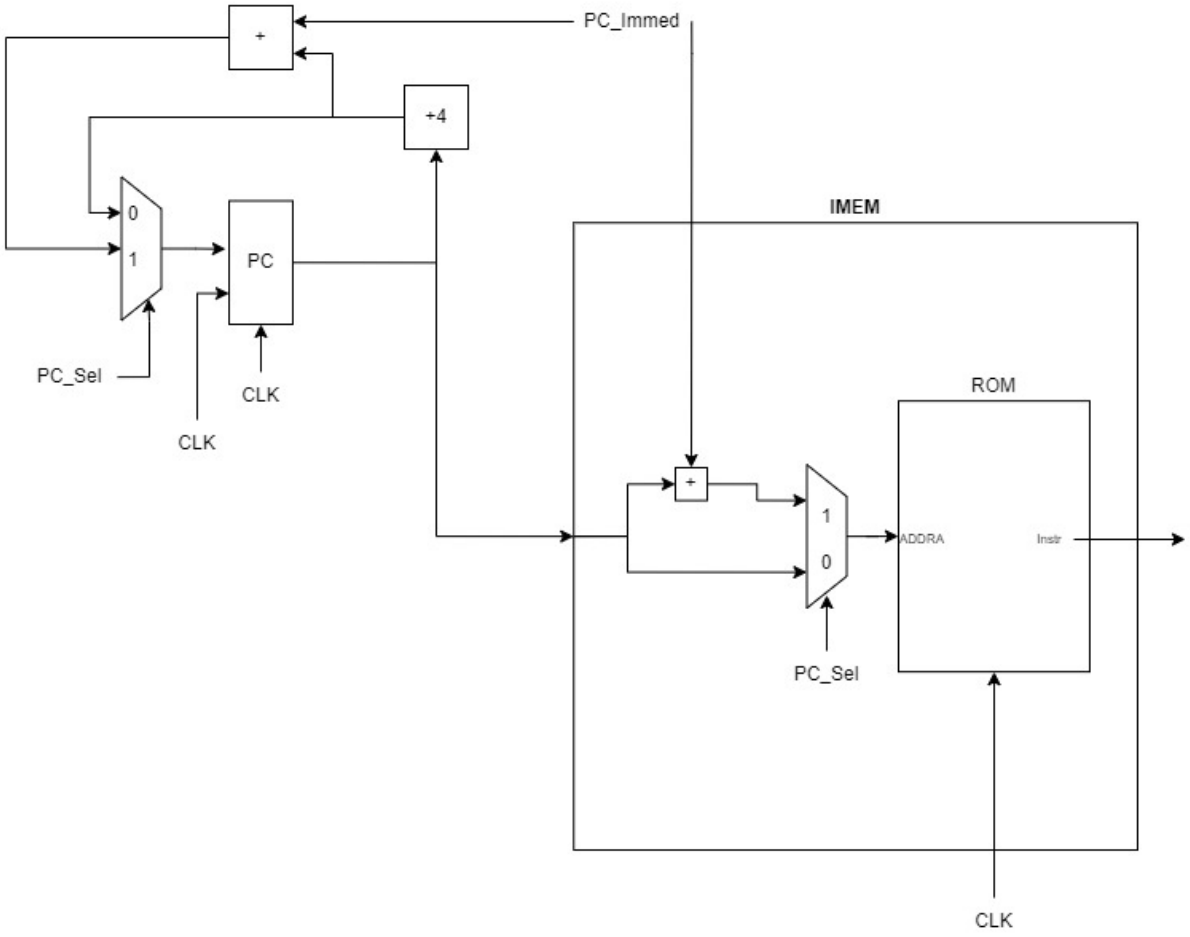
Αναφορά 1ης Φάσης - Οργάνωση Υπολογιστών

Αθανασάκης Ευάγγελος 2019030118
Φραγκογιάννης Γεώργιος 2019030039
Ομάδα Χρηστών 26

Διευκρινίσεις σχετικά με την εκτέλεση της άσκησης:

1. Για την σωστή εκτέλεση τόσο των απλών εντολών (όπου ο PC register προχωράει κατά +4) όσο και των εντολών διακλάδωσης, οι οποίες απαιτούν σωστό υπολογισμό του PC, δημιουργήσαμε παραπάνω λογικά components από αυτά που δίνονται στο σχήμα της εκφώνησης. Οι συνδέσεις που πραγματοποιήθηκαν στο IF STAGE είναι οι εξής:



Εκτός από τη σχεδίαση που περιγράφεται στην άσκηση, εφαρμόζουμε μέσα στο στοιχείο IMEM έναν πολυπλέκτη ο οποίος θα ορίζει την είσοδο της ROM σε περίπτωση "απλής" εντολής η είσοδος της ROM θα είναι η τιμή του PC register ο οποίος θα δείχνει πάντα στην επόμενη γραμμή του προγράμματος. Σε περίπτωση που γίνεται branch η είσοδος της ROM θα πρέπει να γίνει PC+4

+ PC_Immed. Εφόσον το PC+4 θα είναι η έξοδος του PC στο clock που έχει αναγνωστεί η εντολή διακλάδωσης χρειάζεται μόνο να προσθέσουμε το ήδη υπολογισμένο immediate (από το InstrToImmed) για να αναγνωστεί η σωστή επόμενη εντολή από τη ROM. Όσον αφορά την λειτουργία και τις τιμές που θα παίρνει ο PC register ακολουθούμε την εκφώνηση της εργασίας.

2. Για την αρχικοποίηση των εντολών, ακολουθήθηκαν οι συμβάσεις της αρχιτεκτονικής συνόλου εντολών CHARIS όπως δίνονται στην εκφώνηση της άσκησης. Πιο συγκεκριμένα, το format:

6-bits	5-bits	5-bits	5-bits	5-bits	6-bits
Opcode	Rs	Rd	rt	not-used	Func

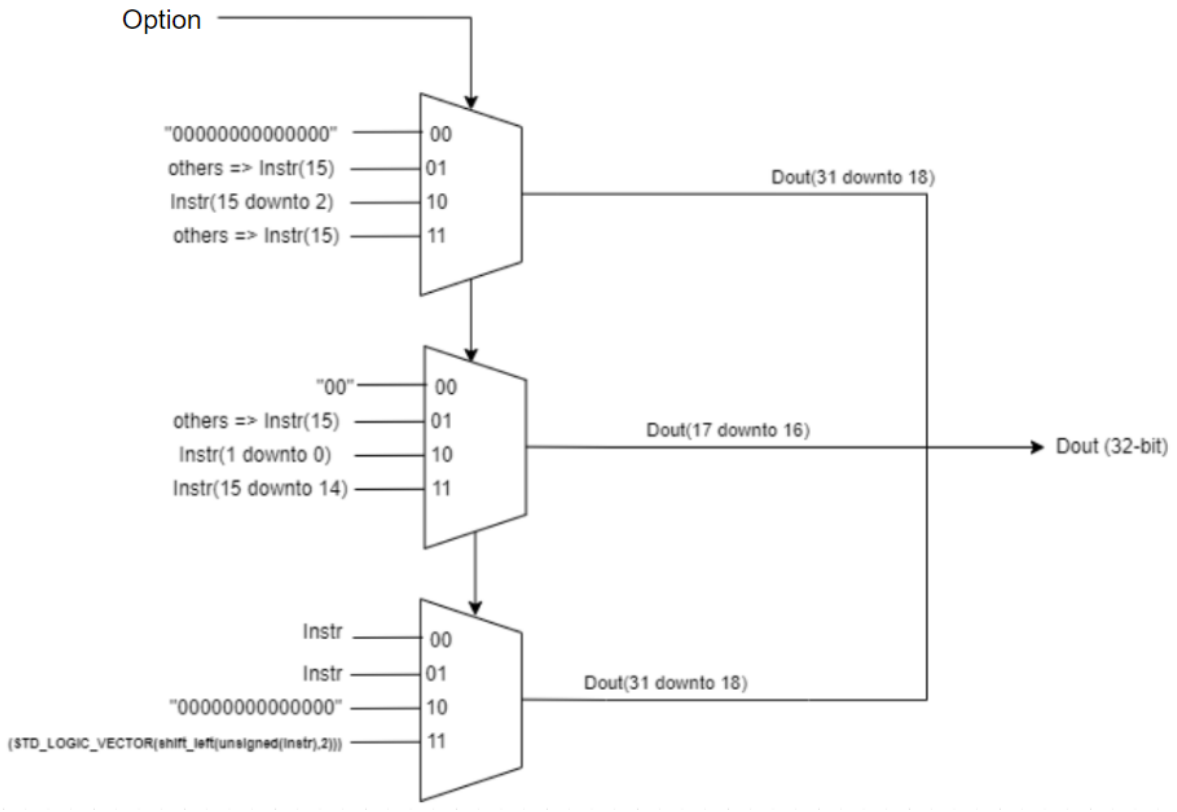
6-bits	5-bits	5-bits	16-bits
Opcode	Rs	rd	Immediate

Επομένως, μια εντολή add R1, R2, R3 => R1=R2+R3 γίνεται:
100000 00011 00001 00010 00000 110000 όπου έχουμε
Opcode : 100000
Rs : 00011 -> R3
Rd : 00001 -> R1
rt : 00010 -> R2
not-used : 00000
func : 110000

3. Για τα opcodes (και τους ανάλογους κώδικες func) χρησιμοποιήθηκαν οι κωδικοί του CHARIS.
4. Εφαρμόστηκε καθυστέρηση στην έξοδο της ALU κατά 10 ns για να αποφευχθεί το Race Condition.
5. Η έξοδος του compare module όταν πληρούνται οι συνθήκες για να είναι ενεργοποιημένη, δηλαδή γράφουμε και διαβάζουμε από τον ίδιο καταχωρητή (ο οποίος δεν είναι ο R0), γίνεται ‘1’ μόνο για 10 ns και έπειτα επιστρέφει στην τιμή ‘0’. Η συγκεκριμένη υλοποίηση έγινε με αυτόν τον τρόπο γιατί παρατηρήθηκε το εξής πρόβλημα. Σε περιπτώσεις εντολών οι οποίες διαβάζουν τιμή από έναν καταχωρητή, εκτελούν πράξεις και έπειτα αποθηκεύουν την τιμή στον ίδιο καταχωρητή (π.χ ror r6, r6, 1), η έξοδος ALU_out αλλάζει και έρχεται στην είσοδο του RF (write data). Επειδή η τιμή του compare module παραμένει στο ‘1’ υπολογίζεται ασύγχρονα η τιμή RF_A και ταυτόχρονα υπολογίζεται ασύγχρονα ξανά η τιμή ALU_out. Το compare module είναι ξανά ενεργοποιημένο εφόσον είμαστε στην ίδια εντολή και αυτήν την φορά έχουμε τα νέα δεδομένα στην είσοδο του RF έτσι σε έναν κύκλο ρολογιού πραγματοποιείται η ίδια εντολή πολλές φορές. Για αυτό, αφήνουμε

την έξοδο του compare module στο '1' μόνο για 10 ns έτσι ώστε όταν έρθει η δεύτερη είσοδος απο την ALU το compare module να είναι απενεργοποιημένο και να μην εμφανιστεί η νέα, όχι επιθυμητή τιμή στην έξοδο.

6. Για τον σωστό υπολογισμό της τιμής Immed ανάλογα με τον κάθε τύπο εντολών, υλοποιήθηκε η βαθμίδα InstrToImmed η οποία παίρνει σαν είσοδο τα 16 LSB του instruction δηλαδή την ποσότητα Immediate των I-type εντολών και υπολογίζει με την βοήθεια ενός flag "Option" του οποίου η τιμή δίνεται απο το control ανάλογα με τις απαιτήσεις του κάθε τύπου εντολής (*00 -> zero-fill ,01 -> sign-extend ,10 -> 16-bit shift,11 -> sign extend and shift 2*) την 32 bit εξοδο.



7. Για τον περαιτέρω έλεγχο της ορθής λειτουργίας του επεξεργαστή εκτελέσαμε το παρακάτω πρόγραμμα:

```
lui R1, 4
sll R1, R1
li R2, 15
add R3, R1, R2
sw R3, 8(R0)
lb R4, 8(R0)
lw R5, 8(R0)
srl R1 ,R1
beq R4, R5, 0
b -2
```

```
111001_00000_00001_0000000000000100
100000_00001_00001_00000_00000_111001
111000_00000_00010_0000000000001111
100000_00001_00011_00010_00000_110000
011111_00000_00011_0000000000001000
000011_00000_00100_0000000000001000
001111_00000_00101_0000000000001000
100000_00001_00001_00000_00000_111000
010000_00100_00101_1111111111110111
111111_00000_00000_1111111111111110
```