Οργάνωση Υπολογιστών - 8η Άσκηση
Ονόμα: Αναγνωστάκης Εμμανουήλ
ΑΜ: 3618

(1)

sample_jump.asm

```
addi $s0,$0,0
addi $s1,$0,4
addi $s2,$0,22
addi $s3,$0,100
addi $s4,$0,1000

main: sw $s3,0($s0)
      add $s0,$s0,$s1
      add $s3,$s3,$s2
      beq $s3,$s4,exit
      j main
exit:
```

sample_just R-Format.asm

```
addi $s0,$0,-10
addi $s1,$0,5
addi $s2,$0,3

main: add $s3,$s1,$s1
      add $s4,$s1,$s0
      sub $s5,$s1,$s2
      sub $s6,$s2,$s1
      and $s7,$s2,$s1
exit:
```

sample_negative,slt.asm

```
addi $s0,$0,0
addi $s1,$0,4
addi $s2,$0,2
addi $s3,$0,-12
addi $s4,$0,10
addi $s6,$0,1

main: add $s3,$s3,$s2
      sw $s3,0($s0)
      add $s0,$s0,$s1
      slt $s5,$s3,$s4
      beq $s5,$s6,main
exit:
```

sample_machinecode.asm

```
addi $s0,$0,0
addi $s1,$0,4
addi $s2,$0,2
addi $s3,$0,0
addi $s4,$0,22
```

sample.asm

```
addi $s0,$0,0
addi $s1,$0,4
addi $s2,$0,2
addi $s3,$0,0
addi $s4,$0,22
main: add $s3,$s3,$s2
      sw $s3,0($s0)
      add $s0,$s0,$s1
      beq $s3,$s4,exit
      beq $zero,$zero,main
exit:
```

---

(2) askisi8-2.asm

```
.register $s1 1
.register $s2 2
.register $s3 3
.register $s4 4

main: add $s3,$s2,$s1
      add $s4,$s3,$s2
      sub $s2,$s4,$s3
      sub $s1,$s3,$s2
      and $s2,$s3,$s4
      and $s1,$s2,$s4
      or $s3,$s2,$s1
      or $s4,$s3,$s2
      slt $s1,$s3,$s2
      slt $s2,$s4,$s1
```

Για όλες τις εντολές από τον κύριο έλεγχο έχουμε τις εξής τιμές για τα σήματα:

RegDst 1
Jump 0
Branch 0
MemtoReg 0
MemWrite 0
MemRead 0
ALUSrc 0
ALUop 10
RegWrite 1

Όλα είναι τα ίδια γιατί όλες οι εντολές είναι τύπου R-Format και δουλεύουν με τον ίδιο τρόπο. Αυτά που έρχονται από τον έλεγχο της ALU είναι διαφορετικά γιατί κάθε εντολή εκτελεί διαφορετική πράξη μεταξύ των τελεστέων.

(3) askisi8-3.asm

```
.register $s2 2
main: sw $s2, 0($s0)
      lw $s1, 0($s0)
exit:
```

Στην ALU παρατηρούμε ότι ο ελεγχος της ALU (ALUControl) δίνει 010 αρα κάνει add. Συγκεκριμένα επειδή ο πολυπλέκτης στην δεύτερη είσοδο της ALU είναι 1 γίνεται πρόσθεση με σταθερά. Το ALUop από τον κύριο έλεγχο είναι 00. Είναι ίδια και για τις δύο, γιατί και στις δυο εντολές add (συγκεκριμένα addi) οπως στο προηγούμενο ερώτημα(εντολη R-Format). Αυτό γίνεται γιατί στην διεύθυνση που περιέχεται ως τιμη στο $s0 προσθέτει η σταθερά 0 (immediate).

(4) askisi8-4.asm

```
.register $s0 0
.register $s1 100
.register $s2 010

main: addi $s0, $s0, 5
      andi $s1, $s1, 100
      ori  $s2, $s2, 110
```

Όλα τα σήματα που έρχονται από τον κυρίως ελεγχο είναι ίδια για όλες καθώς όλες είναι I-Format εντολές. Τα σήματα ALUOp που παίρνει η ALU είναι διαφορετικά ανάμεσα σε αυτές τις εντολές γιατί γίνεται πράξη διαφορετική αλλά είναι ίδια με τις αντίστοιχες εντολές R-Format καθώς ο τελεστέος (ο δεύτερος) έρχεται από τον πολυπλέκτη στην δεύτερη είσοδο της ALU που ανάβει 1 (η είσοδος) για να πάρει τη σταθερά. Η διαφορά με τις εντολές R-Format λοιπόν είναι ότι ο δεύτερος τελεστέος εδώ προέρχεται από το αρχείο καταχωρητών και όχι ως σταθερά καταυθείαν από τον κώδικα της εντολής στην Instruction Memory.

(5)

askisi8-5.asm

```
.register $s0 5
.register $s1 5
.register $s2 15

main: beq $s0, $s2, label1
      ; label1
      addi $s0, $s1, 10
      beq $s0, $s1, main
      addi $s1, $s0, 15

label1: beq $s0, $s1, main
```

Στην αρχή η εντολή beq $s0, $s2, label1 η αφαίρεση της ALU δίνει -10 (απολυτη branch) συνεπώς η AND1 με εισόδους 1 (Branch) και 0 (zero) δίνει 0 και θα περάσει κανονικά δοπιόν από το Mux2 (και το Mux5 αφού δωσ'χουμε jump) η διεύθυνση PC+4 και θα συνεχίσουμε στην από κάτω εντολή. Μετά στην ; label1 έχουμε jump οπότε το σήμα ελέγχου είναι 1 στον Mux5 οπότε ου και να γίνει θα περάσει στον PC η τιμή της διεύθυνσης του label1 βαλι το PC+4 από το Mux2. (τιμή ελέγχου 0 γιατί δεν έχουμε branch). Στο label1 έχουμε beq $s0, $s1, main συνεπώς στην ALU γίνεται αφαίρεση με αποτέλεσμα 0 (επιτυχημένη branch) έχουμε σήματα (branch=1 και zero=1) οπότε η AND1 δίνει 1 και παίρνουμε τη διεύθυνση της main (από την ALUBranch) αντί το PC+4. Στον MUX5 περνάει αυτή η τιμή οπότε η τιμή του PC θα είναι 0.