Μιχάλης Μπαστάκης HY-225 8η Σειρά.
csd 4406

8.7)

a) Το ~~μέρος~~ πρώτο της εντολής addi μοιάζει με αυτό των πράξεων R-to-R. Καθώς διαφέρουν μόνο σε ένα bit (I format) 0010011
(R format) 0110011

Επίσης το funct3 ~~είναι~~ ~~το~~ ~~ίδιο~~ ~~για~~ που έχει η addi (010) set if less than immidiate με το funct3 (010) set if less than (slt)

Οπότε η εντολή addi πρέπει να διαβάσει ένα καταχωρητή και ένα Immidiate να γίνουν οι κατάλληλες πράξεις στον ALU και να αποθηκεύσουν το αποτέλεσμα σε ένα καταχωρητή.

Εφόσον οι opcodes διαφέρουν κατά ένα bit είναι γειτονικοί έτσι έχουν απλοποιήσεις στον χάρτη karnaugh αυτό συμβαίνει γιατί έχουν παρόμοιες λειτουργίες η διαφορά είναι ότι οι εντολές I format διαβάζουν έναν Immidiate αντί για register οπότε θα πρέπει να προσθέσουμε μόνο στις εντολές αν μπορούμε σε ένα κύκλο να διαβάσουμε ~~ένα~~ Immidiate και να κάνουμε πράξεις. Αλλιώς αν πρέπει να πάρουμε ένα κύκλο για να αποθηκευτεί ο Imm και μετά να κάνουμε πράξεις, πρέπει να προσθέσουμε και στο datapath και στον έλεγχο.

β)

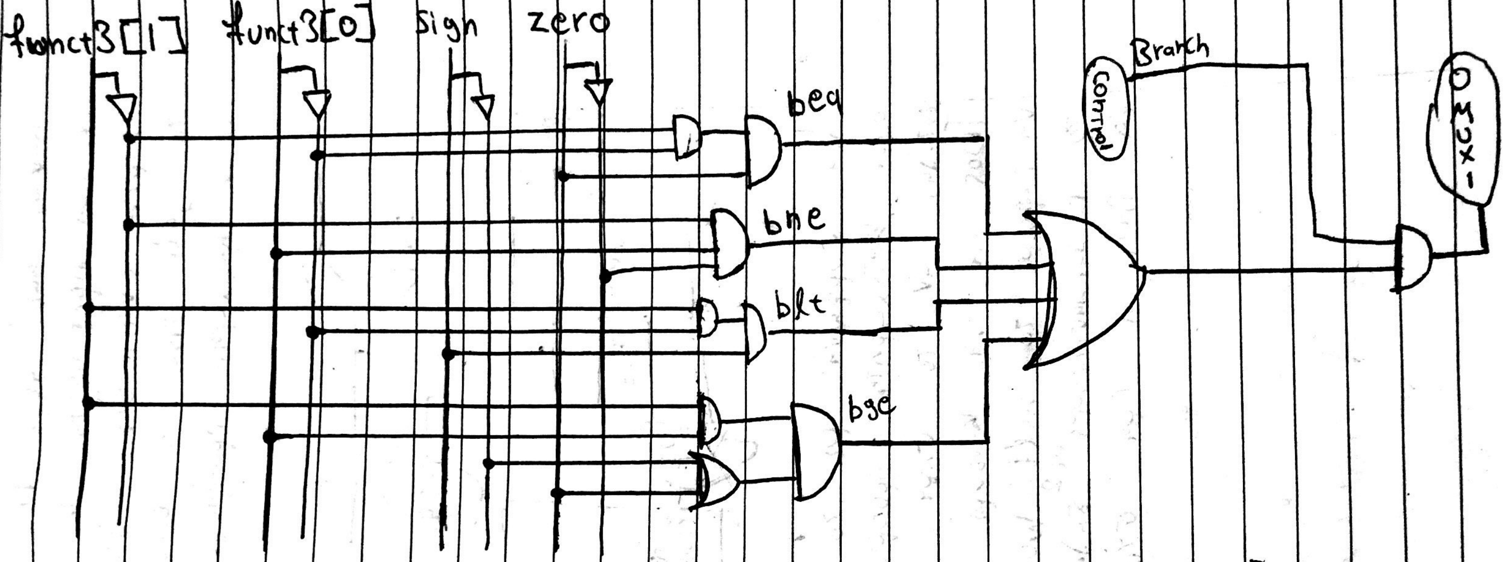
Signal Name	I[6]	I[5]	I[4]	I[3]	I[2]	I[1]	I[0]
Input/[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
add:	0	0	1	0	0	1	1

Signal Name	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop1	ALUOp0
Output								
addi	1	1	1	0	0	0	0	0

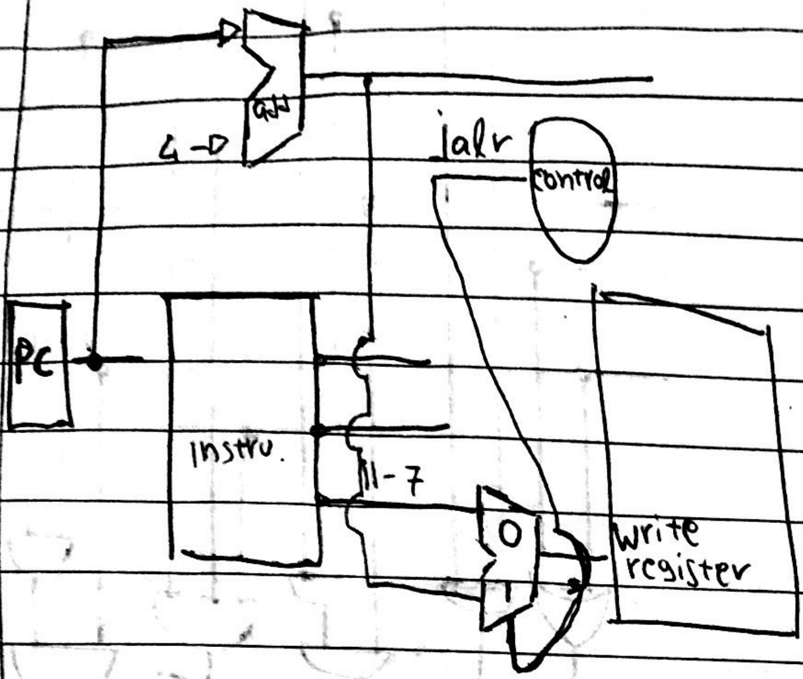
γ) Για να προσθέσουμε τις εντολές bne, blt, bge πρέπει να χρησιμοποιήσουμε τον opcode της branch σε R-format και να αλλάξουμε το funct3. Αφού η branch έχει μια εντολή τώρα την beq με funct3 (000) οι άλλες θα έχουν

funct3	funt3[2]	funct3[1]	funct3[0]
bne	0	0	1
blt	0	1	0
bge	0	1	1

Οπότε χρειαζόμαστε τα 2 LS bits του funct3
Το κύκλωμα θα παραμείνει ίδιο απλά στην ALU πρέπει να προσθέσουμε το sign το οποίο δίνει 1 αν ο αριθμός είναι αρνητικός και 0 αν είναι θετικός. Και για το control στο Branch αντικαθιστούμε την AND με το εξής.


funct3[1]
funct3[0]
sign
zero
beq
bne
blt
bge
Control
Branch
MUX1

8.7)
δ) Η τιμη PC+4 υπάρχει στο datapath ηδη καθως καθε φορα που το PC δινει τιμη αυτη η τιμη προστιθεται με το 4 για να παψε στη επόμενη εντολη. Οπότε όταν το control δει οτι το jalr είναι τα opcodes που πηρε βαζει στον καταχωρητη προορισμου αυτην την τιμη



Άρα συμφωνα με το πάνω σχήμα αποθηκευουμε στον καταχωρητη προορισμου την τιμη της επομενης εντολής αν το σημα jalr είναι ανοικτο.
Τέλος για να κάνουμε jump στο imm πρέπει να περουμε το half word απο το read data να το πολλαπλάσιασουμε $^{με 2}$ δηλαδη shift left 1 και να πάει η τιμη στον PC

8.8)

a) IM ~~Control~~
500 ps ~~+ 200 ps~~ = ~~700 ps~~

b) IM Control
500 + 200 = 700 ps

c) IM
500 + 300 = 800 ps

d) 500 + 300 + 400 = 1200 ps

e) 400 ps

f) 400 ps

g) 500 + 300 + 400 + 500 = 1700 ps

h) 1700 ps

i) 1700 ps + 200 ps = 1900 ps

Η μεγαλύτερη καθυστέρηση ειναι 1900 ps αρα 1,9 ns   T = 1,9 ns

$$f = \frac{1}{T} = \frac{1}{1,9 \cdot 10^{-9}}$$ [illegible]

$$= 0,526 \; 10^{9} = 526 \text{ MHz}$$

Εφοσον τώρα θέλουμε 4400 και πριν ειχαμε 400 η διαφορα 4400 - 400 = 4000 ps η διαφορ.

$$f = \frac{1}{4 \cdot 10^{-9}} = 250 \text{ MHz}$$

άρα 776 MHz $\frac{526}{776} =$