Οργάνωση Υπολογιστών

Μιχαήλ Μαρκετάκης

2017030165

Αναφορά Εργασίας 2

Επεξεργαστής πολλαπλών κύκλων(Multi-cycle Processor)

Μετατροπή του Datapath ενός κύκλου σε Datapath πολλαπλών κύκλων

Για την δημιουργία του επεξεργαστή πολλαπλών κύκλων έγιναν οι παρακάτω αλλαγές στο DATAPATH,βασιζόμενοι στο μοντέλο του single-cycle επεξεργαστη.

• Αρχικά τοποθετήσαμε 5 νέους καταχωρητές,μεταξύ των stages.Πιο συγκεκριμένα οι νέοι καταχωρητές είναι οι εξής:

Instruct_REG (Καταχωρητής για την Εντολή που βγαίνει από το text Segment της μνήμης)

R_A (Καταχωρητής για την πρώτη τιμή που διαβάζει το RF)

R_B (Καταχωρητής για την δεύτερη τιμη που διαβάζει το RF)

R_ALU (Καταχωρητής για το αποτέλεσμα της πράξης της ALU)

R_MEM(Καταχωρητής για την εξοδο της μνήμης (Εντολές lw,lb))

• Έγιναν μετατροπές στο κομμάτι του IFSTAGE του single cycle επεξεργαστή,αφού πλέον η μετακίνηση του Programm Counter κατά +4 ή SignedExtend(Imm<<2) θα γίνεται πλέον μέσω της ALU.

Πιο Συγκεκρίμενα:

PC +4 θα γίνεται μέσω της εξόδου της ALU(ALU_OUT)

PC +4 +SignedExtend(Imm <<2) θα γίνεται μέσω του της εξοδου του καταχωρητή της ALU(R_ALU) δηλαδή **ALU_regOut**.

• Στο κομμάτι του EXSTAGE προσθέθηκε ένας πολυπλέκτης 2 προς 1 ,όπου παίρνει σαν πρώτη είσοδο την έξοδο του R_A καταχωρητή(**A_regOUt**),καθώς επίσης και την έξοδο του καταχωρητή PC(**PC_OUT**),με την έξοδο του να αποτελεί την πρώτη είσοδο της ALU,αναλόγως την τιμή του σήματος επιλογής **ALU_SrcA**.

Τέλος,προσθέθηκε ακόμη ενας πολυπλέκτης,4 προς 1 αυτή την φορά,ο οποίος στην πραγματικότητα αντικατέστησε τον ήδη υπάρχον πολυπλέκτη του Exstage.Ο νέος αυτός πολυπλέκτης δέχεται σαν είσοδο:

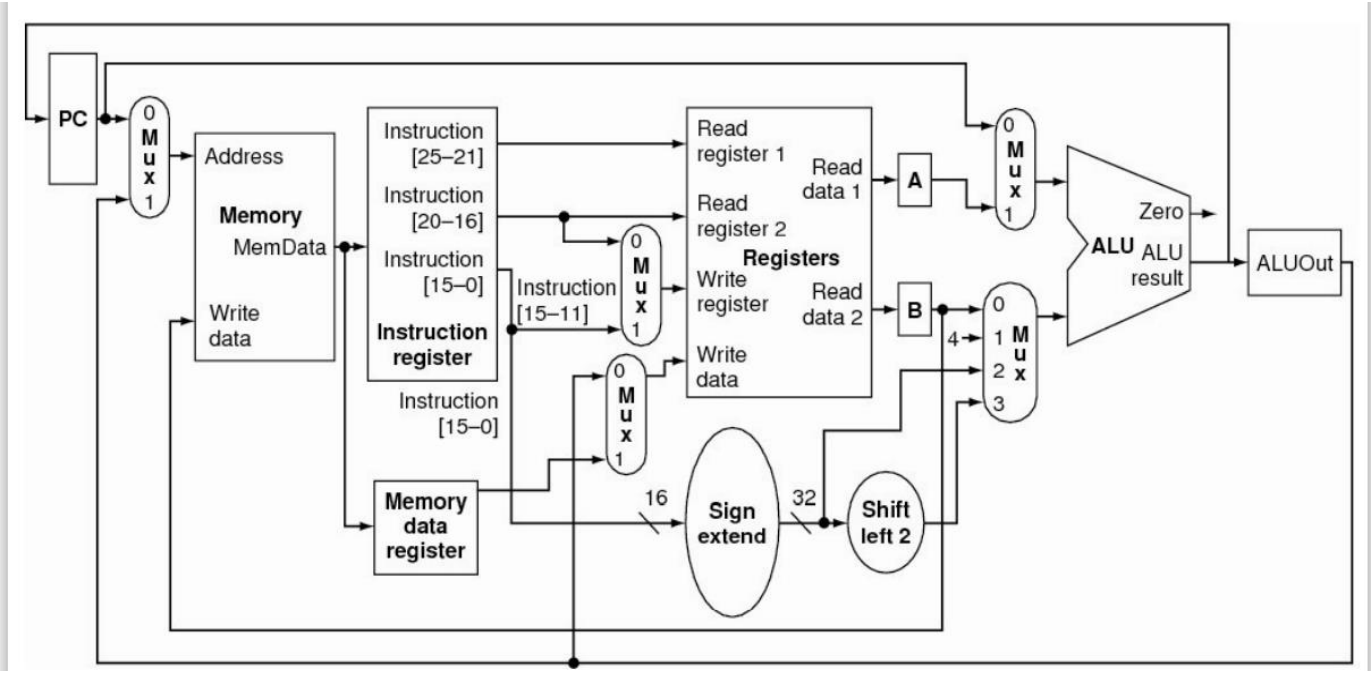
•Την έξοδο του καταχωρητή R_B(Rt ή Rd)

•Το 4(Για την Μετακίνηση του PC)

•Το Immed (32-bit) ύστερα από τις αλλαγές που υποβλήθει στο <<συννεφάκι>>

Και με βάση την τιμή του σήματος επιλογής **ALU_SrcB**,η έξοδος αυτού του πολυπλέκτη αποτελεί την δεύτερη είσοδο της ALU.

Στο παρακάτω διάγραμμα φαίνεται ακριβώς η Αρχιτεκτονική υλοποίησης του Multi-Cycle Datapath.



Σχεδιασμός και Υλοποίηση του Control

Το control μέρος της υλοποίησης είχε αρκετά διαφορετική δομή σε σχέση με το αντίστοιχο του Single Cycle επεξεργαστή.

Υλοποιήθηκε ως μία Μηχανή Πεπερασμένων Καταστάσεων(FSM).

Ο λόγος που χρησιμοποιήσαμε FSM ήταν γιατί σε κάθε κατάσταση θέλαμε να ενεργοποιούμε διαφορετικά σήματα.

Η FSM μας πάντα δίνοντας Reset ξεκιναει από την κατασταση **Start**.Η Start είναι η κατάσταση που διαβάζει την εκάστοτε εντολή(σειριακά),αυξάνει δηλαδη την τιμή του Program Counter κατά 4,και επανερχόμαστε σε αυτή μετά το τέλος κάθε εντολής.

Η κατάσταση **Start1** είναι μια ενδιάμεση κατάσταση,όπου διαδέχεται την Start,και προφανως όλες οι εντολές περνάνε και από αυτήν.Η ιδιαιτερότητα αυτής της κατάστασης είναι ότι αποκωδικοποιεί το OpCode της Εντολής,και αναλόγως την τιμή του η FSM <<μεταπηδά>> στο αντίστοιχη κατάσταση.

Αξίζει να σημειώσουμε,την λειτουργικότητα της κατάστασης **BBFlagStage**,η οποία χρησιμεύει ώστε να αυξήσει την τιμή του Program Counter κατά SignedExtend(Immed<<2) (Εντολές Beq,Bne)

Ανάλογα τα στάδια του datapath που χρησιμοποιεί κάθε εντολή καθως επίσης λαμβάνοντας υπόψη τις καθυστερήσεις,δημιουργήσαμε καποιον αριθμό από καταστάσης δίνοντας τα κατάλληλα σήματα για την εκάστοτε εντολή,ώστε να ολοκληρωθεί επιτυχώς.

Τέλος,παρατηρήσαμε ότι οι καταστάσεις της FSM μας εξαρτούνται και από την είσοδο(συγκεκριμένα από το opcode της κάθε εντολής),άρα θα χαρακτηρίζαμε την μηχανή πεπερασμένων καταστάσεων ως **MEALY**.

Γνωριζουμε ότι ο Single Cycle επεξεργαστής ήταν αρκετά αργός,αφου είχε πολύ μεγάλο κύκλο ρολογιού, και ο κύκλος αυτός προσδιοριζόταν από τον χρόνο που απαιτούσε η πιο αργή εντολή(Load) .Για τις υπόλοιπες εντολές εν ολίγοις είχαμε σπατάλη χρόνου,αφού η εντολή θα είχε τελείωσε και θα περιμέναμε μέχρι την επόμενη ακμή του ρολογιού.

Η υλοποίηση του Multi-cycle επεξεργαστή έγινε αφού πλέον η κάθε εντολή καταλαμβάνει όσο χρόνο χρειάζεται πραγματικά με αποτέλεσμα να κάνουμε ουσιαστικά τον επεξεργαστή μας πιο γρήγορο.

Όπως είπαμε και παραπάνω κάθε εντολή οργανώνεται σε μικρότερα steps(καταστάσεις FSM) και εκτελούμε το κάθε step μέσα σε έναν κύκλο,με

αποτέλεσμα να μειώσουμε σημαντικά τον κύκλο του ρολογιού.

Για μεγαλύτερη βεβαίωση σχετικά με την λειτουργικότητα του επεξεργαστή μας,εκτος από το πρόγραμμα αναφοράς ένα και δύο,δημιουργήθηκαν ακόμα 3 προγράμματα αναφοράς,τα οποία περιέχουν όλες τις εντολές που δεν υπάρχουν στα προηγούμενα,καθώς και συνδιασμούς αυτών.

• Πρόγραμμα Αναφοράς 1 (MC_TEST1) :

bne $3 , $3 , 8 --failed ----> (PC + 4)

b -2 --back to bne,infinite loop!

addi $1 , $0 , $1 -- never execute

• Πρόγραμμα Αναφοράς 2 (MC_TEST2):

addi $3,$0,2 -- addi 2 in register 3

beq $1,$1,2 -- true--->(PC+4+signedExtend(Immed<<2))

ori $3,$0,0xABCD -- never execute

sw $3 , 4($zero) --never execute

sb $3 , 8($zero) -- store byte from register 3 in position 1026

• Πρόγραμμα Αναφοράς 3 (MC_TEST3):

```
li $6, 7

lui $5, 4

li $4, 7

Add $10 ,$6 , $5

sw $10, 16($zero)

beq $6, $4,1  -- true ( PC+4+SingedExtend(immed<<2)

lw $1,16($zero)  -- never execute

Add $2,$zero,$4
```