Οργάνωση και Σχεδίαση Υπολογιστών (ΗΥ232)
Χειμερινό Εξάμηνο 2022-2023

Εργαστήριο 5

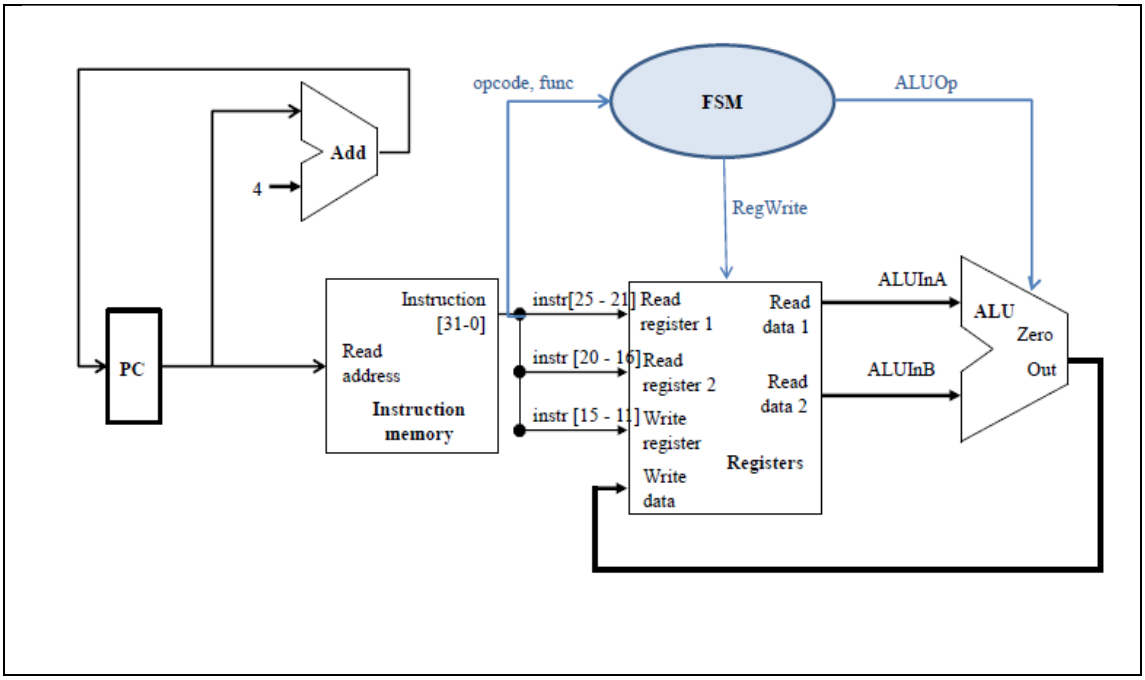
1. Υλοποίηση του MIPS (format R) ενός κύκλου μηχανής (5 μονάδες)

Ο στόχος του εργαστηρίου είναι η υλοποίηση και επαλήθευση του τμήματος δεδομένων (datapath unit) και του τμήματος ελέγχου (control unit) του επεξεργαστή MIPS **ενός κύκλου μηχανής**, χρησιμοποιώντας την γλώσσα περιγραφής υλικού Verilog, και την βιβλιοθήκη έτοιμων μονάδων, μέρος της οποίας υλοποιήσατε στο προηγούμενο εργαστήριο. Στο πρώτο τμήμα του εργαστηρίου θα υλοποιήσετε μερικές από της εντολές MIPS που ακολουθούν το format R, και συγκεκριμένα τις εντολές **add**, **sub**, **or**, **and** και **slt**. Η θεωρία για το εργαστήριο αυτό καλύπτεται από τα Κεφ. 4.3-4.4 του βιβλίου.

Η εικόνα 1 δείχνει το διάγραμμα της αρχιτεκτονικής που θα πρέπει να υλοποιήσετε στην πρώτη φάση αυτού του εργαστηρίου, χρησιμοποιώντας τις δύο μονάδες ALU και Register File του προηγούμενου εργαστηρίου. Ο PC είναι ο 32-bit Program Counter και δείχνει την θέση στην Instruction Memory από την οποία θα πρέπει να διαβαστεί η επόμενη εντολή. Ο PC θα πρέπει να γίνει σωστά reset όπως εξηγήσαμε και στην τάξη στην τιμή 32'h0 για να είμαστε σε θέση να εκκινήσουμε το πρόγραμμα.

Επιπλέον σας δίδεται και η υλοποίηση μιας μνήμης 1 KB η οποία και μπορεί να χρησιμοποιηθεί για την αποθήκευση εντολών MIPS 32-bits. Στις παρακάτω παραγράφους περιγράφουμε την δομή και λειτουργία της μνήμης.

Το τμήμα ελέγχου (control unit) οδηγεί το κάθε σήμα εισόδου του τμήματος δεδομένων (data



Εικόνα 1. Αρχιτεκτονική MIPS ενός κύκλου μηχανής για εντολές R-format.

```
@0  0110_4020  // HEX code for add $t0, $t0, $s0
@1  0124_4822  // HEX code for  sub $t1, $t1, $a0
@2 ….
```

path), και αντίστροφα, το κάθε σήμα εξόδου του τμήματος δεδομένων οδηγεί το τμήμα ελέγχου. Έτσι, συνενώνοντας τα δυο τμήματα, προκύπτει η πλήρης υλοποίηση του επεξεργαστή. Το τμήμα ελέγχου είναι σχετικά απλό σε αυτό το εργαστήριο μιας και χρειάζεται να δημιουργήσει πολύ λίγα σήματα ελέγχου, λαμβάνοντας σαν είσοδο τα κομμάτια της 32-bit εντολής *opcode* και *func* που απαιτούνται για την αποκωδικοποίηση της. Θα ήταν προτιμότερο να τοποθετήσετε την μονάδα ελέγχου σε διαφορετικό αρχείο από το τμήμα δεδομένων για να διευκολυνθείτε στην υλοποίηση αργότερα. Μια καλή ιδέα θα ήταν, για παράδειγμα, να έχετε και ένα αρχείο *cpu.v* σε υψηλότερο επίπεδο αφαίρεσης (abstraction) το οποίο και θα τοποθετεί το module του τμήματος δεδομένων και το module του τμήματος ελέγχου και θα τα συνδέει.

Μονάδα μνήμης - Memory

`module Memory (clock, reset, ren, wen, addr, din, dout);`
Ασύγχρονη μνήμη εντολών/δεδομένων.

Θύρες:

clock: Είσοδος. Ρολόι συγχρονισμού. Όλες οι εγγραφές σε καταχωρητές θα πρέπει να λαμβάνουν χώρα στην αρνητική ακμή του ρολογιού (negative edge), όπως ακριβώς και στο αρχείο καταχωρητών. Το clock δημιουργείται στο testbench.

reset: Είσοδος. Ασύγχρονο reset. Active low. Μόνο όταν το reset είναι 1 λειτουργεί κανονικά η μνήμα. Το σήμα reset δεν χρησιμοποιείται για να μηδενίσει τα στοιχεία της μνήμης. Το reset δημιουργείται στο testbench.

ren (Read Enable): Είσοδος (1 bit): ενεργοποίηση ανάγνωσης.

wen (Write Enable): Είσοδος (1 bit): ενεργοποίηση εγγραφής.

addr: Είσοδος (32 bits): διεύθυνση προσπέλασης. Η διεύθυνση αναφέρεται σε bytes. Δεδομένου ότι εμείς έχουμε μόνο λέξεις στην υλοποίησή μας, τα δύο χαμηλότερα bit της διεύθυνσης θα πρέπει να είναι μηδέν. Για πρακτικούς λόγους, κατά την προσομοίωση, υλοποιούνται μόνο οι πρώτες 1024 λέξεις της μνήμης. Επομένως, τα 20 υψηλότερα bits της διεύθυνσης δεν χρησιμοποιούνται και είναι don't cares.

din (Data Input): Είσοδος (32 bits): είσοδος δεδομένων προς εγγραφή.

dout (Data Output): Έξοδος (32 bits): έξοδος δεδομένων ανάγνωσης.

Αρχικοποίηση της Μνήμης

Η αρχικοποίηση της μνήμης μπορεί να γίνει με τις εντολές **$readmemh** ή **$readmemb** της Verilog η οποία διαβάζει δεδομένα από ένα αρχείο και αρχικοποιεί πίνακες με αυτά τα δεδομένα. Παράδειγμα χρήσης της **$readmemh** στο πλαίσιο δοκιμής μέσα σε κάποια δομή **initial** στο testbench είναι το εξής:

```
$readmemh("program.hex", cpu0.cpu_IMem.data);
```

Όπου `cpu0` είναι το όνομα της εμφάνισης της μονάδας cpu του επεξεργαστή, `cpu_IMem` το όνομα της εμφάνισης της μνήμης του επεξεργαστή και `data` είναι το όνομα του πίνακα της μνήμης στην μονάδα μνήμης της βιβλιοθήκης. Το σχετικό αρχείο μνήμης, `program.hex` είναι ένα *text file* και πρέπει να βρίσκεται στον ίδιο κατάλογο με την εκτέλεση της προσομοίωσης και να έχει την παρακάτω μορφή:

```
@0  0110_4020  // HEX code for add $t0, $t0, $s0
@1  0124_4822  // HEX code for  sub $t1, $t1, $a0
@2  …
```

Η πρώτη στήλη, @X, αντιστοιχεί στην διεύθυνση της μνήμης X, ενώ η δεύτερη στήλη στα δεδομένα που θα αποθηκευτούν κατά την αρχικοποίηση στην διεύθυνση X. Προσέξτε ότι η διεύθυνση της μνήμης πρέπει πάντα να είναι εκφρασμένη στο δεκαεξαδικό σύστημα. Η κάτω παύλα (underscore) είναι διαχωριστικός χαρακτήρας της Verilog, για την ευκολότερη ανάγνωση των δεδομένων. Αν χρησιμοποιηθεί η εντολή **$readmemb**, τα δεδομένα θα πρέπει να είναι σε δυαδικό σύστημα. Για παράδειγμα:

```
@0  0000_0001_0001_0000_0100_0000_0010_0000
@1  …
```

Σημείωση: Μπορείτε να αναπτύξετε τον κώδικα σας στον MARS, να κάνετε assemble και να χρησιμοποιήσετε το "*Dump Memory to File*" εικονίδιο για να σώσετε τον κώδικα σε κάποιο text αρχείο σε HEX ή BIN format. Αυτό θα σας βοηθήσει για να παράγετε το *program.hex* αρχείο αυτόματα.

Προσομοίωση για Επαλήθευση Ορθής Λειτουργίας

Μαζί με την δομική περιγραφή του τμήματος δεδομένων του επεξεργαστή, θα πρέπει να υλοποιηθεί και ένα πλαίσιο ελέγχου, το οποίο θα εκτελεί ένα μικρό πρόγραμμα αποθηκευμένο στην μνήμη, το οποίο θα επαληθεύει της ορθή λειτουργία εκτέλεσης για τους διαφορετικούς τύπους εντολών, δηλ. τύπου R.

Για την επαλήθευση του τμήματος δεδομένων του επεξεργαστή παρέχετε ένα πρότυπο σκελετού πλαισίου δοκιμής, το οποίο θα πρέπει να τροποποιήσετε (αρχείο *testbench.v*). Στο πρότυπο αυτό αρχείο εμπεριέχονται οδηγίες για την εμφάνιση της μονάδας του επεξεργαστή, την αρχικοποίηση του καταχωρητή, τον ορισμό του ρολογιού και την εφαρμογή των σημάτων ελέγχου ανά κύκλο. Η αρχικοποίηση του αρχείου καταχωρητών θα γίνει με απευθείας ανάθεση της τιμής σε κάθε καταχωρητή, ενώ η αρχικοποίηση της μνήμης εντολών θα γίνει με την χρήση της εντολής **$readmemh**.

Η επαλήθευση του κυκλώματος σας θα πρέπει να γίνεται με το να ελέγχετε τις τιμές των καταχωρητών σε κάθε κύκλο μηχανής. Σύμφωνα με όσα έχουμε πει, κάθε εντολή διαρκεί ακριβώς έναν κύκλο μηχανής, και η μόνη μονάδα που μπορεί να μεταβληθεί από την εκτέλεση του προγράμματος είναι το αρχείο καταχωρητών.

2. Υλοποίηση του MIPS (LW, SW, Branches) ενός κύκλου μηχανής (5 μονάδες)

Στο δεύτερο τμήμα του εργαστηρίου θα ολοκληρώσετε την υλοποίηση του MIPS ενός κύκλου μηχανής, ενσωματώνοντας τις εντολές **lw, sw**, **addi**, **beq**, και **bne** που ακολουθούν το format I. Η θεωρία για το εργαστήριο αυτό καλύπτεται από τα Κεφ. 4.3-4.4 του βιβλίου.

Η εικόνα 2 δείχνει το διάγραμμα της αρχιτεκτονικής που θα πρέπει να υλοποιήσετε, εμπλουτίζοντας το τμήμα δεδομένων (datapath unit) και το τμήμα ελέγχου (control unit) της αρχιτεκτονικής του προηγούμενου εργαστηρίου.

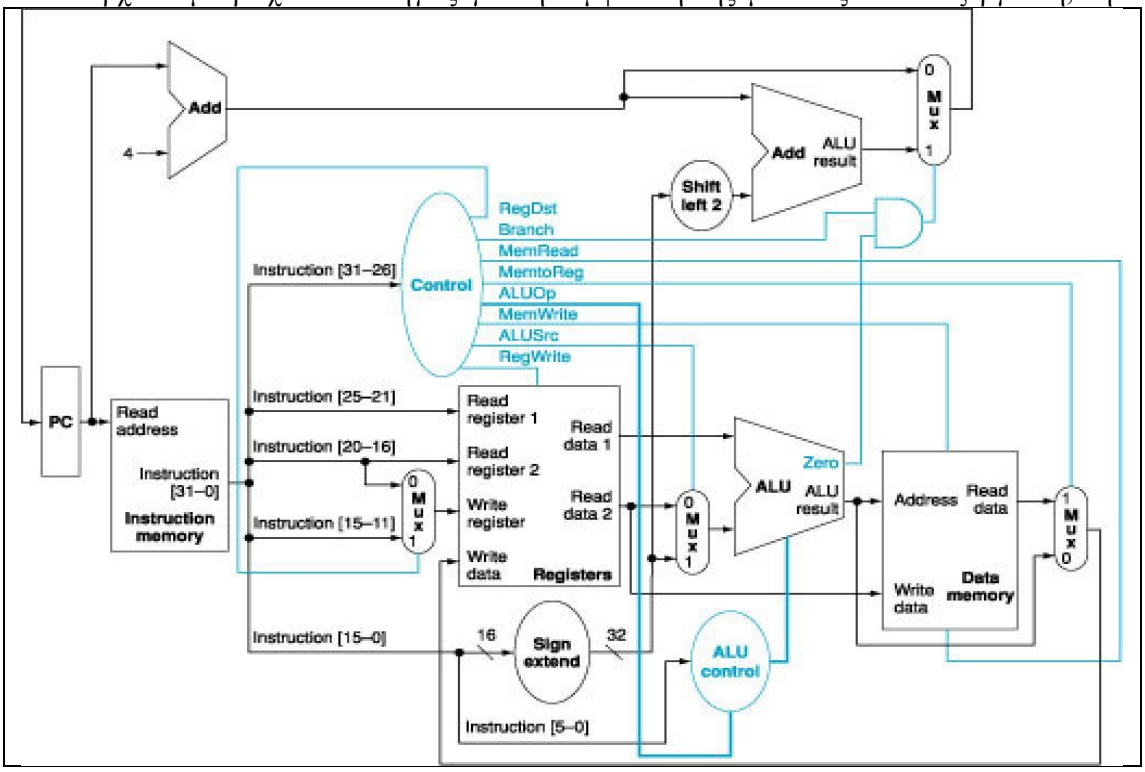
Ιδιαίτερη προσοχή θα πρέπει να δώσετε στην σωστή συνδεσμολογία των υπομονάδων. Για παράδειγμα η μνήμη δεδομένων (data memory) θα χρησιμοποιηθεί από τις εντολές **lw** και **sw**, η μονάδα πρόσθεσης **ADD** για την εύρεση της νέας διεύθυνσης του PC μετά από μια εντολή branch, καθώς και ο μεγάλος αριθμός από πολυπλέκτες (multiplexers) όπως φαίνεται και στο σχήμα.

Προσομοίωση για Επαλήθευση Ορθής Λειτουργίας

Μαζί με την δομική περιγραφή του τμήματος δεδομένων του επεξεργαστή, θα πρέπει να υλοποιηθεί και ένα πλαίσιο ελέγχου, το οποίο θα εκτελεί ένα μικρό πρόγραμμα αποθηκευμένο στην μνήμη, το οποίο θα επαληθεύει της ορθή λειτουργία εκτέλεσης για τους διαφορετικούς τύπους εντολών . Το προτεινόμενο πρόγραμμα φαίνεται παρακάτω.
Μπορείτε να χρησιμοποιήσετε διαφορετικό πρόγραμμα για την επαλήθευση, αρκεί να επαληθεύσετε όλους τους τύπους εντολών.

```
label: add $t1, $t0, $a0
       sw $t1, 5($t3)
       lw $s2, 5($t3),
       slt $t1, $t0, $s3
       addi $s0, $s0, 1
       beq $t1, $at, label
```

Εάν έχετε όρεξη και χρόνο θα μπορούσατε να υλοποιήσετε και επιπλέον εντολές που ακολουθούν το I-format, όπως για παράδειγμα τις εντολές του τύπου **subi, lb, sb**, κοκ. Για την επαλήθευση του τμήματος δεδομένων του επεξεργαστή παρέχετε ένα πρότυπο σκελετού πλαισίου δοκιμής, το οποίο θα πρέπει να τροποποιήσετε (αρχείο *testbench.v*). Στο πρότυπο αυτό αρχείο εμπεριέχονται οδηγίες για την εμφάνιση της μονάδας του επεξεργαστή, την



Εικόνα 2. Αρχιτεκτονική MIPS ενός κύκλου μηχανής για εντολές R-format, load/store και branches .

αρχικοποίηση του καταχωρητή, τον ορισμό του ρολογιού και την εφαρμογή των σημάτων ελέγχου ανά κύκλο. Η αρχικοποίηση του αρχείου καταχωρητών θα γίνει με απευθείας ανάθεση της τιμής σε κάθε καταχωρητή, ενώ η αρχικοποίηση της μνήμης εντολών θα γίνει με την χρήση της εντολής **$readmemh**.

Η επαλήθευση του κυκλώματος σας θα πρέπει να γίνεται με το να ελέγχετε τις τιμές των καταχωρητών σε κάθε κύκλο μηχανής. Σύμφωνα με όσα έχουμε πει, κάθε εντολή διαρκεί ακριβώς έναν κύκλο μηχανής.