ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

# Αρχιτεκτονική Υπολογιστών

## Εργαστηριακή Άσκηση 3 (50% εργαστηριακού βαθμού)
## Ομαδική εργασία σε ομάδες τριών (3) μελών
## Ημερομηνία παράδοσης: 11/6/2023, 23:59 (μέσω e-class)

Σε αυτήν την Εργαστηριακή Άσκηση θα κατασκευάσετε έναν RISC pipelined CPU. Ήδη στις προηγούμενες ασκήσεις έχετε κατασκευάσει αρκετά από τα μέρη τα οποία θα χρησιμοποιεί το CPU σας. Σε αυτήν την άσκηση θα κατασκευάσετε τα υπόλοιπα και κάνοντας χρήση αυτών και των προηγούμενων, θα συνθέσετε τον ζητούμενο επεξεργαστή. Πριν ξεκινήσετε την εργασία μελετήστε προσεκτικά την 6η σειρά διαλέξεων (06 - MIPS Datapath) και βεβαιωθείτε ότι καταλαβαίνετε όλα τα παραδείγματα.

Τα κομμάτια που πρέπει να υλοποιήσετε είναι αναλυτικά τα ακόλουθα:

## Καταχωρητές διοχέτευσης

### Register IF_ID

Λαμβάνει ως είσοδο το PC (από τον PC Register) και το Instruction (από το Input) καθώς και τα clock και mode (flush/enable) και επιστρέφει το PC **(i) στην ALU** (που το χρησιμοποιεί μέσω πολυπλέκτη 2 σε 1 αν η εντολή είναι MFPC) καθώς και **(ii) ξανά πίσω στον PC Register**. Αν το enable είναι 1 τότε το PC πηγαίνει στην επόμενη εντολή (+2 -16-bit) ενώ αν είναι flush τότε μηδενίζονται τα περιεχόμενα του PC και του outInstruction.

### Register ID_EX

Λαμβάνει είσοδο από τον Controller (τον τύπο της εντολής), το αρχείο καταχωρητών και τον SignExtender και δρομολογεί το output τόσο στην ALU, π.χ. αν πρόκειται για R-Type εντολή ή στον EX_MEM Register, π.χ. αν πρόκειται για LW/SW ή PrintDigit – έξοδος οθόνης.

### Register EX_MEM

Λαμβάνει ως είσοδο τα PrintDigit, ReadDigit, WriteEnable, isLW, από τον ID_EX Register, το clock από την είσοδο, το Result από την ALU και τα RegAD και R2Reg από τον ID_EX Register. Εκχωρεί τις εισόδους στην έξοδο στο rising edge. Το RegAD_EXMEM πάει στον Forwarder και στο Register File και το Result τόσο στην έξοδο όσο και πίσω την είσοδο.

### Register MEM_WB

Λαμβάνει ως είσοδο το αποτέλεσμα (είτε από το keyboard –keyData- αν είναι το isReadDigit του controller αληθές) είτε από τη μνήμη –fromData- αν το isLW του controller είναι 1, καθώς εοπίσης και τη διεύθυνση του καταχωρητή (RegAD) και επιστρέφει τις ίδιες τιμές στο rising edge. Εκχωρεί τιμές στα σήματα writeData και writeAD, που πηγαίνουν στο RegisterFile, στα αντίστοιχα write ports.

## Μονάδα προώθησης (forwarder)

Ελέγχει τις διευθύνσεις (του αρχείου καταχωρητών) που πρέπει να γραφτεί κάποιο αποτέλεσμα, αλλά βρίσκονται ακόμα στα στάδια Memory ή Write Back και πρέπει και να διαβαστούν παράλληλα τα δεδομένα τους. Εάν δηλαδή ο καταχωρητής που διαβάσαμε είναι ίδιος με το καταχωρητή ενός από αυτά τα στάδια, θα πρέπει να κάνουμε το forwarding. Αλλιώς θα διαβάσουμε τα παλιά δεδομένα. Προφανώς όταν μιλάμε για τον καταχωρητή 0, η εγγραφή δεν πραγματοποιείται, συνεπώς δεν πρέπει να κάνουμε το forwarding. Δείτε επίσης το παράδειγμα της διαφάνειας 95 (06 - MIPS Datapath).

## Επιλογέας (Select)

Ανάλογα με την απόφαση του Forwarder διαβιβάζει την κατάλληλη τιμή από το αρχείο καταχωρητών, τη μνήμη, ή το WriteBack στην ALU σαν πρώτη παράμετρο. Χρησιμοποιείται επίσης και για τις δύο εισόδους της ALU.

## Μονάδα κινδύνου (Hazard Unit)

Η μονάδα κινδύνου αποφασίζει για το εάν θα πρέπει να γίνει flush η εντολή. Flush γίνεται μια εντολή όταν έχει προηγηθεί ένα branch ή ένα jump, οπότε η τρέχουσα εντολή πρέπει να διαγραφεί. Δείτε επίσης το παράδειγμα της διαφάνειας 101. Επίσης αποφασίζει για το εάν η επόμενη εντολή θα πρέπει να γίνει flush. Τέλος αποφασίζει για το ποιο σήμα θα πρέπει να φορτωθεί στον program counter.

## Μονάδα παγίδευσης (Trap Unit)

Παρακολουθεί το opcode κάθε εντολής και μόλις δεχθεί την εντολή End Of Running αναγκάζει το σύστημα σε ένα διαρκές flush εντολών αλλά και σε «πάγωμα» του PC. Με αυτό τον τρόπο αποφεύγεται η εκτέλεση άκυρων εντολών (dummy instructions).

## JRSelector

Πρόκειται για επιλογέα με βάση το JRopcode που λαμβάνει από το HazardUnit και επιλέγει τη διεύθυνση (jump) που θα σταλεί στον PC: Αν το JRopcode έχει την τιμή 00, τότε περνάει το PCP2ADD όπως έχει υπολογιστεί από τον IF_ID Register (+2 – επόμενη διεύθυνση), αν έχει την τιμή 01 περνάει το JumpAD, αν έχει την τιμή 10 περνάει το BranchAD και σε κάθε άλλη περίπτωση περνάει το PCP2AD.

## Program Counter (PC) Register

Στέλνει τη διεύθυνση εκτέλεσης της τρέχουσας εντολής ή jump ή branch στον IF_ID register (που με τη σειρά του υπολογίζει την επόμενη εντολή (+2) και τη στέλνει στην ALU) καθώς και στην έξοδο (instructionAD).

## Control

Ελέγχει την διαδικασία εκτέλεσης των εντολών και παράγει όλα τα σήματα ελέγχου. Αναλυτικά αποφασίζει για  τα παρακάτω. Προσοχή, αάν η εντολή πρέπει να γίνει flush, τότε όλες οι παραπάνω τιμές μηδενίζονται, οπότε σε όλα αυτά το control θα δίνει απάντηση 0 εάν πρέπει να γίνει flush η εντολή.

- Εάν πρόκειται για την εντολή MFPC (move from PC)
- Εάν πρόκειται για εντολή Jump
- Εάν πρόκειται για την εντολή διαβάσματος ψηφίου
- Εάν πρόκειται για την εντολή εκτύπωσης στην οθόνη
- Τον καταχωρητή που θα αποθηκευτεί η πληροφορία-αποτέλεσμα
- Εάν πρόκειται για εντολή τύπου R
- Εάν πρόκειται για την εντολή save word
- Εάν πρόκειται για την εντολή load word
- Εάν πρόκειται για εντολή branch
- Εάν πρόκειται για την εντολή Jump Register

## Επεξεργαστής

Θα πρέπει χρησιμοποιώντας τα κυκλώματα που έχετε φτιάξει να συνθέσετε το CPU και να κάνετε το simulation στο Quartus. Το RTL διάγραμμα του επεξεργαστή φαίνεται αναλυτικά στην τελευταία σελίδα.

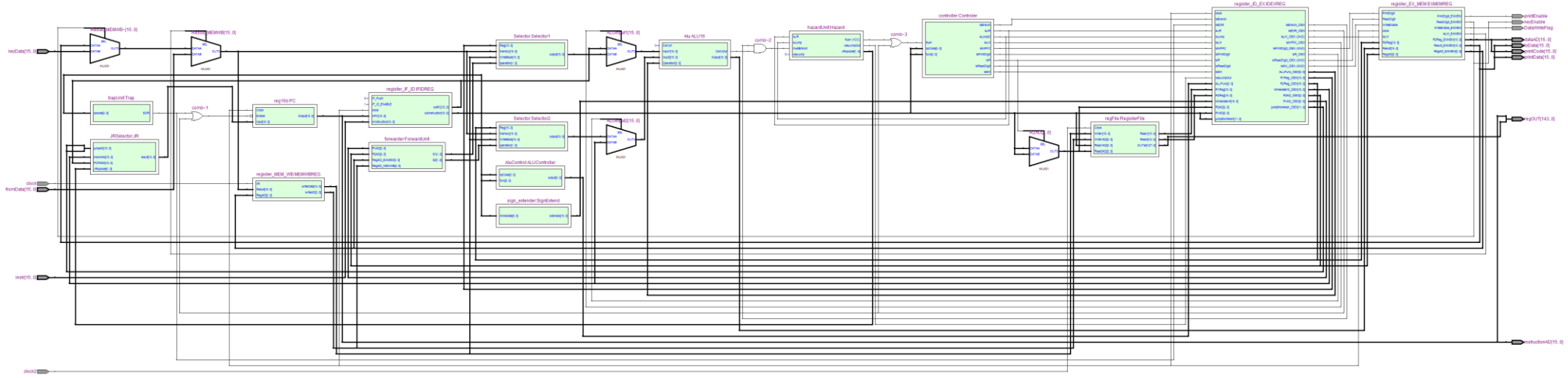
## Ονοματοδοσία

Για να μην υπάρξει σύγχυση με την ονοματοδοσία, παρατίθενται παρακάτω κάποιοι κανόνες, τους οποίους είναι ισχυρά συνιστώμενο αλλά όχι υποχρεωτικό να ακολουθήσετε όσο περισσότερο μπορείτε.

| | **Ερμηνεία** | **Παράδειγμα** |
|---|---|---|
| `is*` | Τα σήματα που έχουν boolean ερμηνεία | **`isR`**: Σήμα που φέρει την πληροφορία του εάν η εντολή είναι τύπου R |
| `*AD` | Η διεύθυνση του αντίστοιχου καταχωρητή | **`reg2AD`**: Η διεύθυνση του 2ου καταχωρητή |
| `IF,ID,EX,ME,WB` | Τα ονόματα των αντίστοιχων σταδίων του pipelining | **`regADME`**: Η διεύθυνση του καταχωρητή που έρχεται από το στάδιο Memory |

## Γενικές Οδηγίες

- Στα σχεδιαγράμματα, το νούμερο πάνω από την πλάγια γραμμή δηλώνει τον αριθμό των bits για το αντίστοιχο σήμα. Όπου δεν υπάρχει, εννοείται ότι μιλάμε για σήμα του ενός bit.
- Χρησιμοποιήστε ίδια ονόματα για σήματα, modules κλπ με αυτά που σας δίνονται.
- Μπορείτε να χρησιμοποιήσετε είτε behavioral είτε structural κώδικα, αλλά ενδείκνυται ο behavioral όπου αυτό είναι δυνατόν.
- Για όλα τα προβλήματα πραγματοποιήστε το ανάλογο functional και timing simulation για το πρέπον διάστημα προκειμένου να επαληθεύσετε την σωστή λειτουργία του κυκλώματός σας.
- Να έχετε σύντομα και περιεκτικά σχόλια στην VHDL όταν χρειάζεται να εξηγήσετε «δύσκολα» κομμάτια κώδικα, ρουτίνες, κλπ.
- Γράψτε (σε σχόλια) στην αρχή κάθε αρχείου τα **ονόματά σας και ΑΜ**.
- Καλείται ένας φοιτητής από κάθε ομάδα να αναρτήσει την εργασία σε αρχείο της μορφής ergasia3_ΑριθμόςΜητρώουΜέλους1_ΑΜΜέλους2_ΑΜΜέλους3.zip (που όπως βλέπετε δεν θα πρέπει να είναι .rar).


keyData[15..0]
clock
fromData[15..0]
inst[15..0]
clock2
trapUnit:Trap
JRSelector:JR
reg16b:PC
register_IF_ID:IFIDREG
forwarderForwardUnit
Selector:Selector1
Selector:Selector2
AluControl:ALUController
sign_extender:SignExtend
Alu:ALU16
hazardUnitHazard
controller:Controller
regFile:RegisterFile
register_ID_EX:IDEXREG
register_EX_MEM:EXMEMREG
printEnable
readEnable
DataWriteFlag
dataIn[15..0]
dataOut[15..0]
printCode[15..0]
printData[15..0]
regOUT[143..0]
instructionAD[15..0]