ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Αρχιτεκτονική Υπολογιστών

Εργαστηριακή Άσκηση 2 (25% εργαστηριακού βαθμού)
Ομαδική εργασία σε ομάδες τριών (3) μελών
Ημερομηνία παράδοσης: 14/5/2023, 23:59 (μέσω e-class)

Στη δεύτερη εργασία θα κατασκευάσετε κάποια από τα επιμέρους κομμάτια από τα οποία θα αποτελείται η CPU που θα συνθέσουμε στην επόμενη εργασία. Η CPU αυτή έχει πάρα πολλά κοινά στοιχεία με αυτή που περιγράφεται στο βιβλίο «Οργάνωση και Σχεδίαση Υπολογιστών», όμως έχει και διαφορές, με πιο σημαντική διαφορά το ότι μιλάμε για **16bit** αρχιτεκτονική και όχι 32bit.

Τα κομμάτια που καλείστε να υλοποιήσετε είναι τα εξής:

- Καταχωρητής (reg)
- Ψευδοκαταχωρητής 0 (reg0)
- Πολυπλέκτης 8 σε 1 (mux8to1)
- Αποκωδικοποιητής 3 σε 8 (decoder3to8)
- Αρχείο Καταχωρητών (reg_file)
- Επέκταση Immediate (sign_extender)
- Υπολογισμός JumpAddress (jump_add)
- ALU Control (ALU_control)
- ALU(ALU)

Αναλυτική Περιγραφή

Καταχωρητής (reg)

Πρόκειται για ένα σύνολο από D-Flip Flop. Είναι ουσιαστικά παραλλαγή του 1β ερωτήματος της πρώτης εργαστηριακής άσκησης. Το μέγεθος της εισόδου και εξόδου, θα πρέπει να δίνεται παραμετρικά με χρήση `generic`. Θα παίρνει σαν είσοδο το σήμα ρολογιού όπως επίσης και σήμα Enable (για το αν θα επιτρέπεται ή όχι η εγγραφή). Η εγγραφή γίνεται μόνο όταν το σήμα Enable είναι 1, και τη στιγμή που “ανεβαίνει” το ρολόι.

Ψευδοκαταχωρητής 0 (reg0)

Θα προσομοιώνει τη συμπεριφορά ενός καταχωρητή αλλά θα βγάζει σαν έξοδο πάντα 0. Η χρήση της τιμής 0 είναι τόσο συχνή σε εντολές αρχιτεκτονικής υπολογιστών που πολύ συχνά υλοποιούμε έναν καταχωρητή που αποθηκεύει και δίνει πάντα την τιμή 0.

Πολυπλέκτης 8 σε 1 (mux8to1)

Δέχεται σαν είσοδο 8 αριθμούς των 16 bit και τους πολυπλέκει σε έναν, με βάση ένα 3 bit σήμα επιλογής. Για βοήθεια σχετικά με την υλοποίηση του πολυπλέκτη ανατρέξτε στις διαφάνειες του 2ρου εργαστηρίου.

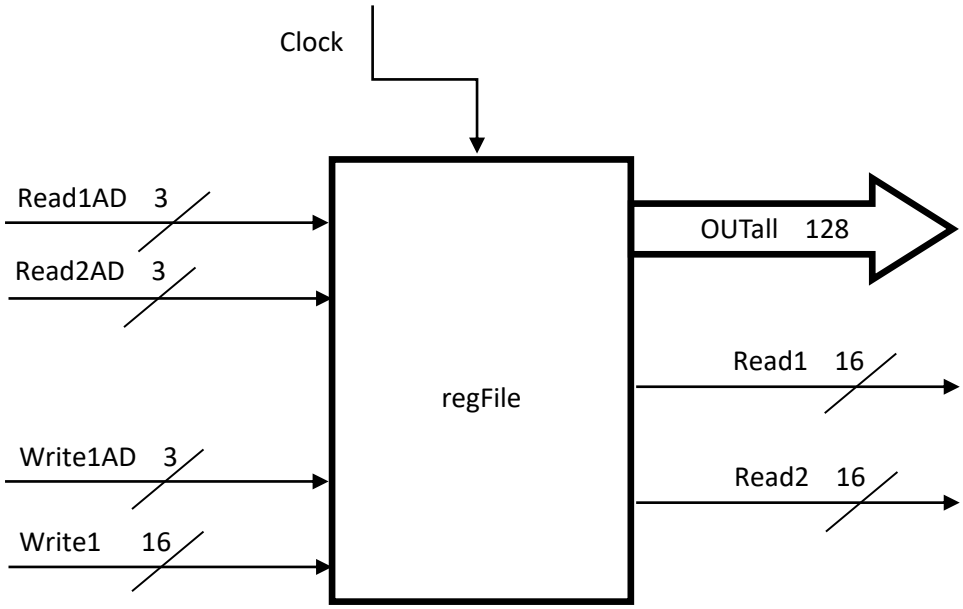
Αποκωδικοποιητής 3 σε 8 (decoder3to8)

Δέχεται σαν είσοδο έναν 3 bit αριθμό και κάνει αποκωδικοποίηση σε 8 σήματα εξόδου. Κατά την αποκωδικοποίηση, θα πρέπει να παράγεται ένα σήμα το οποίο να έχει όλα τα bit του 0 εκτός από το bit που καθορίζεται από τον σήμα εισόδου. Για παράδειγμα το input “010”, θα πρέπει να παράγει το output “00000100”.

Αρχείο Καταχωρητών (reg_file)

Χρησιμοποιώντας τα παραπάνω φτιάξτε ένα αρχείο οκτώ 16 bit καταχωρητών (ενώς ψευδοκαταχωρητή και άλλων 7 καταχωρητών). Θα δέχεται σαν είσοδο δύο διευθύνσεις καταχωρητών των οποίων τα δεδομένα θα πρέπει να εξάγει σε δύο 16 bit εξόδους. Επίσης θα δέχεται σαν είσοδο 16 bit δεδομένα προς εγγραφή καθώς και τη διεύθυνση του καταχωρητή όπου αυτά θα αποθηκευτούν. Επειδή η εγγραφή στον ψευδοκαταχωρητή δεν έχει αλλοίωση, όταν δεν θέλουμε να κάνουμε εγγραφή κάποιας τιμής, επιλέγουμε την διεύθηνση του ψεδοκαταχωρητή.

Θα πρέπει να έχει επίσης σαν έξοδο ένα vector που αποτελείται από τα 128 σήματα των καταχωρητών με τη σειρά με την οποία αριθμούνται (από 0 έως 15 στον καταχωρητή 0, από 16 έως 31 στον καταχωρητή 1, κλπ).



Επέκταση Immediate (sign_extender)

Θα παίρνει ως είσοδο τον 6 bit αριθμό immediate και θα τον επεκτείνει ώστε η έξοδος που θα παραχτεί να είναι 16 bits. Η επέκταση θα γίνεται με τον εξής τρόπο: τα τελευταία 6 bits θα παραμένουν ως έχουν και τα υπόλοιπα 10 bits θα έχουν τιμή του προσήμου του immediate.

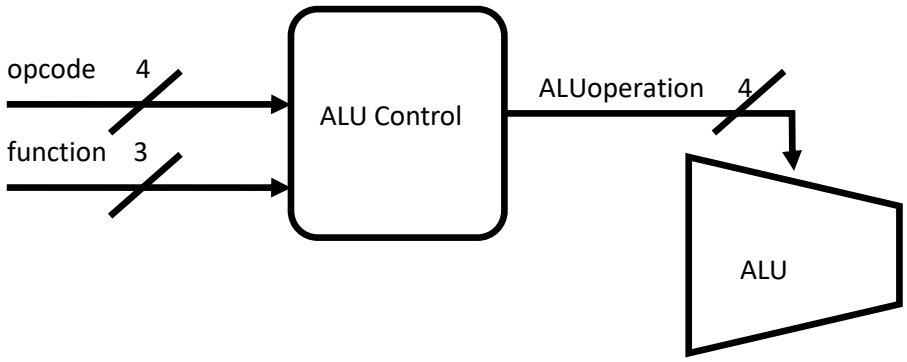
Υπολογισμός JumpAddress (jump_add)

Παίρνει ως είσοδο την τρέχουσα διεύθυνση instrP2AD και κάνει Jump με βάση την είσοδο jumpAD. Για την υλοποίηση της εντολής θα πρέπει να κάνετε επέκταση της εισόδου jumpAD (αντίστοιχα με την επέκταση του sign_extender) και να την διπλασιάσετε (λόγω 16 bit αρχιτεκτονικής), πρωτού την προσθέσετε με την τρέχουσα διεύθυνση.

Ουσιαστικά οι πράξεις που πρέπει να κάνετε μπορούν να εκφραστούν με τον ψευδοκώδικα: `EjumpAD=extend12to16(jumpAD)*2+instrP2AD`

ALU Control (ALU_control)

Το ALU Control αποφασίζει για το ποια πράξη πρέπει εκτελέσει η ALU. Στην περίπτωση που το opcode είναι “0000”, η εντολή που θα εκτελεστεί είναι τύπου R και η κατάλληλη πράξη θα επιλέγεται με βάση την είσοδο function. Στις υπόλοιπες περιπτώσεις η εντολή θα επιλέγεται αποκλειστικά με βάση την είσοδο opcode.



ALU(ALU)

Εάν δεν έχετε πλήρως έτοιμη και λειτουργική την ALU στην πρώτη εργαστηριακή άσκηση (και συγκεκριμένα στο 1α) μπορείτε να την ξαναφτιάξετε με όποιον τρόπο προτιμάτε (όχι μόνο structural κώδικα). Η περιγραφή των πράξεων βρίσκεται στην πρώτη εργασία. Επίσης, επισημαίνεται ότι σε αυτή την περίπτωση, δεν θα γίνει καμία αναβαθμολόγηση της πρώτης εργασίας για όσους την υλοποιήσουν εκπρόθεσμα.

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 2^{ου} καταχωρητή

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 κομμάτια πραγματοποιήστε το ανάλογο simulation για να επαληθεύσετε τη σωστή λειτουργία του κυκλώματός σας.
- Να έχετε σύντομα και περιεκτικά σχόλια στην VHDL όταν χρειάζεται να εξηγήσετε «δύσκολα» κομμάτια κώδικα, ρουτίνες, κλπ.

- Η παράδοση της εργασίας περιλαμβάνει τον VHDL κώδικα, τα αντίστοιχα RTL διαγράμματα καθώς και τα simulations, όπως και στην πρώτη εργασία.
- Γράψτε (σε σχόλια) στην αρχή κάθε αρχείου τα **ονόματά και τους ΑΜ σας**.
- Καλείται ένας φοιτητής από κάθε ομάδα να αναρτήσει την εργασία σε αρχείο της μορφής **ergasia2_ΑριθμόςΜητρώουΜέλους1_ΑΜΜέλους2_ΑΜΜέλους3.zip** (που όπως βλέπετε δεν θα πρέπει να είναι .rar).