ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Αρχιτεκτονική Υπολογιστών

Εργαστηριακή Άσκηση 2 (25% εργαστηριακού βαθμού)

Ημερ/νία παράδοσης: 30 Μαΐου 2021, 23.55

Στο δεύτερο lab θα κατασκευάσετε κάποια από τα επιμέρους κομμάτια από τα οποία θα αποτελείται το CPU που θα συνθέσουμε σε επόμενη άσκηση.

Το CPU αυτό έχει πάρα πολλά κοινά στοιχεία με αυτό 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 πρέπει να φτιάξετε είναι τα εξής:

>Καταχωρητής (reg)
>Ψευδοκαταχωρητής 0(reg0)
>Πολυπλέκτης 8 σε1 (mux8to1)
>Αποκωδικοποιητής 3 σε 8(decode3to8)
>Αρχείο Καταχωρητών(regFile)
>Επέκταση Immediate (sign_extender)
>Υπολογισμός JumpAddress(jumpAD)
>ALU Control(ALUcontrol)
>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 όμως, 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 η εγγραφή). Η εγγραφή γίνεται μόνο όταν το 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.

>Πολυπλέκτης 8 σε 1 (mux8to1)

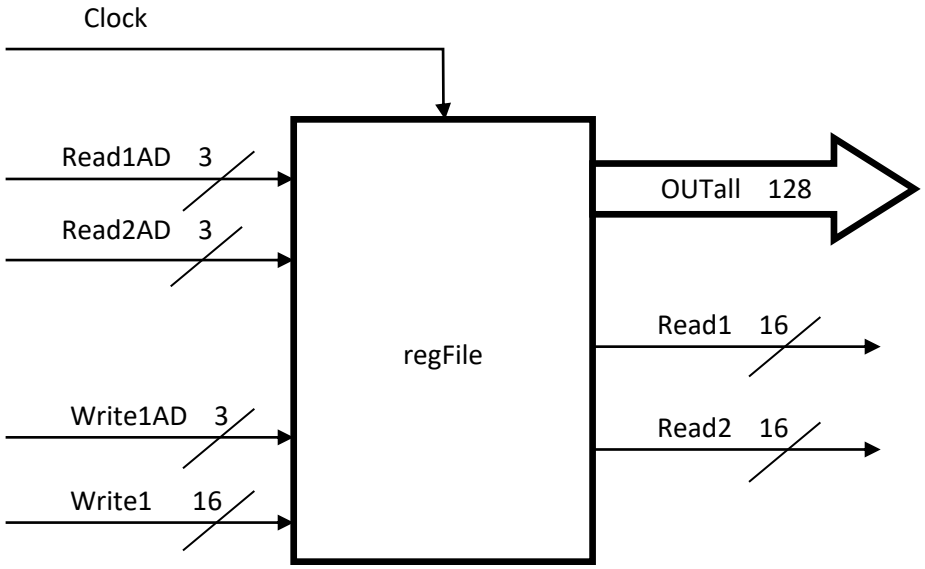
Δέχεται σαν είσοδο 8 αριθμούς των 16 bit και τους πολυπλέκει σε έναν, με βάση ένα 3bito σήμα επιλογής.

>Αποκωδικοποιητής 3 σε 8 (decode3to8)

Δέχεται σαν είσοδο έναν 3-bito αριθμό και κάνει αποκωδικοποίηση σε 8 σήματα εξόδου.

>Αρχείο Καταχωρητών (regFile)

Χρησιμοποιώντας τα παραπάνω φτιάξτε ένα αρχείο 8 16-bitων καταχωρητών (ψευδοκαταχωρητής και άλλοι 7). Θα δέχεται σαν είσοδο 2 διευθύνσεις καταχωρητών των οποίων τα δεδομένα θα πρέπει να εξάγει σε 2 16bites εξόδους. Επίσης θα δέχεται σαν είσοδο 16-bit δεδομένα προς εγγραφή καθώς και τη διεύθυνση του καταχωρητή όπου αυτά θα αποθηκευτούν. Προφανώς η εγγραφή στον καταχωρητή 0 δεν έχει αλλοίωση, και γι' αυτό όταν δεν θέλουμε να γράψουμε πουθενά, «γράφουμε» στον καταχωρητή 0.
Θα πρέπει να έχει επίσης σαν έξοδο ένα vector από 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). Το τελικό αποτέλεσμα φαίνεται στο σχήμα 2.1.



Σχήμα 2.1

>Επέκταση Immediate (sign_extender)

Κάνει επέκταση του πεδίου immediate. Σέβεται το πρόσημο. Το πεδίο immediate είναι 6 bits, και η έξοδος 16 bits.

> Υπολογισμός JumpAddress (jumpAD)

Πρόκειται για την καινούργια διεύθυνση, όταν κάνουμε jump. Υπολογίζεται με βάση την τρέχουσα και την τιμή που λέει η εντολή jump. Γίνεται λοιπόν επέκταση του πεδίου jumpAddr. Σέβεται το πρόσημο. Επίσης προσθέτει το διπλάσιο του αποτελέσματος (λόγω του ότι έχουμε 16-bit αρχιτεκτονική) στην τρέχουσα διεύθυνση.

```
EjumpAD=extend13το16(jumpAD)*2+instrP2AD
```

Σχήμα 2.2

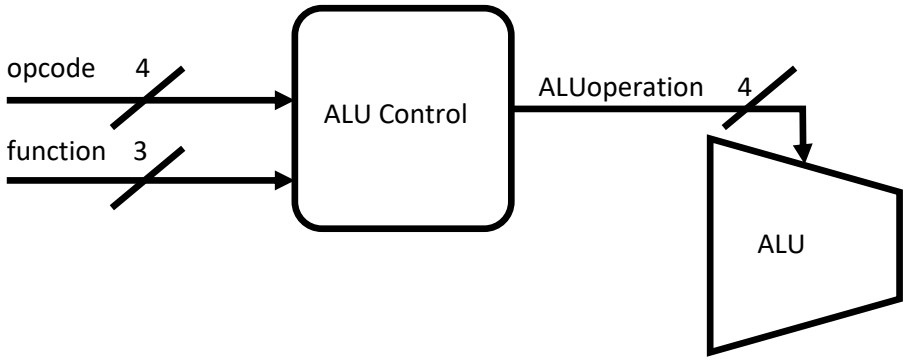
>ALU(ALU)

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 και στην πρώτη εργαστηριακή άσκηση.

>ALU Control (ALUcontrol)

Το ALU Control αποφασίζει για το ποια πράξη πρέπει εκτελέσει η ALU.

Εάν η εντολή είναι τύπου R, τότε θα πρέπει να εκτελέσει την κατάλληλη πράξη, ανάλογα με το πεδίο function της εντολής. Εάν δεν είναι τύπου R, θα πρέπει να εκτελεστεί η ανάλογη πράξη. Μπορούμε να δούμε μια σχηματική αναπαράσταση στην εικόνα 2.3.



Ονοματοδοσία

Για να μην υπάρξει σύγχυση με την ονοματοδοσία, παρατίθενται παρακάτω κάποιοι κανόνες, τους οποίους είναι καλό 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 προβλήματα πραγματοποιήστε το ανάλογο functional και timing simulation για το πρέπον διάστημα 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.
- Γράψτε (σε σχόλια) στην αρχή κάθε αρχείου τα **ονόματά σας και ΑΜ**.
- Η παράδοση της άσκησης περιλαμβάνει τον VHDL κώδικα, τα αντίστοιχα RTL διαγράμματα καθώς και τα simulations. Η παράδοση θα πρέπει να γίνει μέσω του e-class σαν ergasia2.zip από την επιλογή Εργασίες Φοιτητών σύμφωνα με το παρακάτω υπόδειγμα.