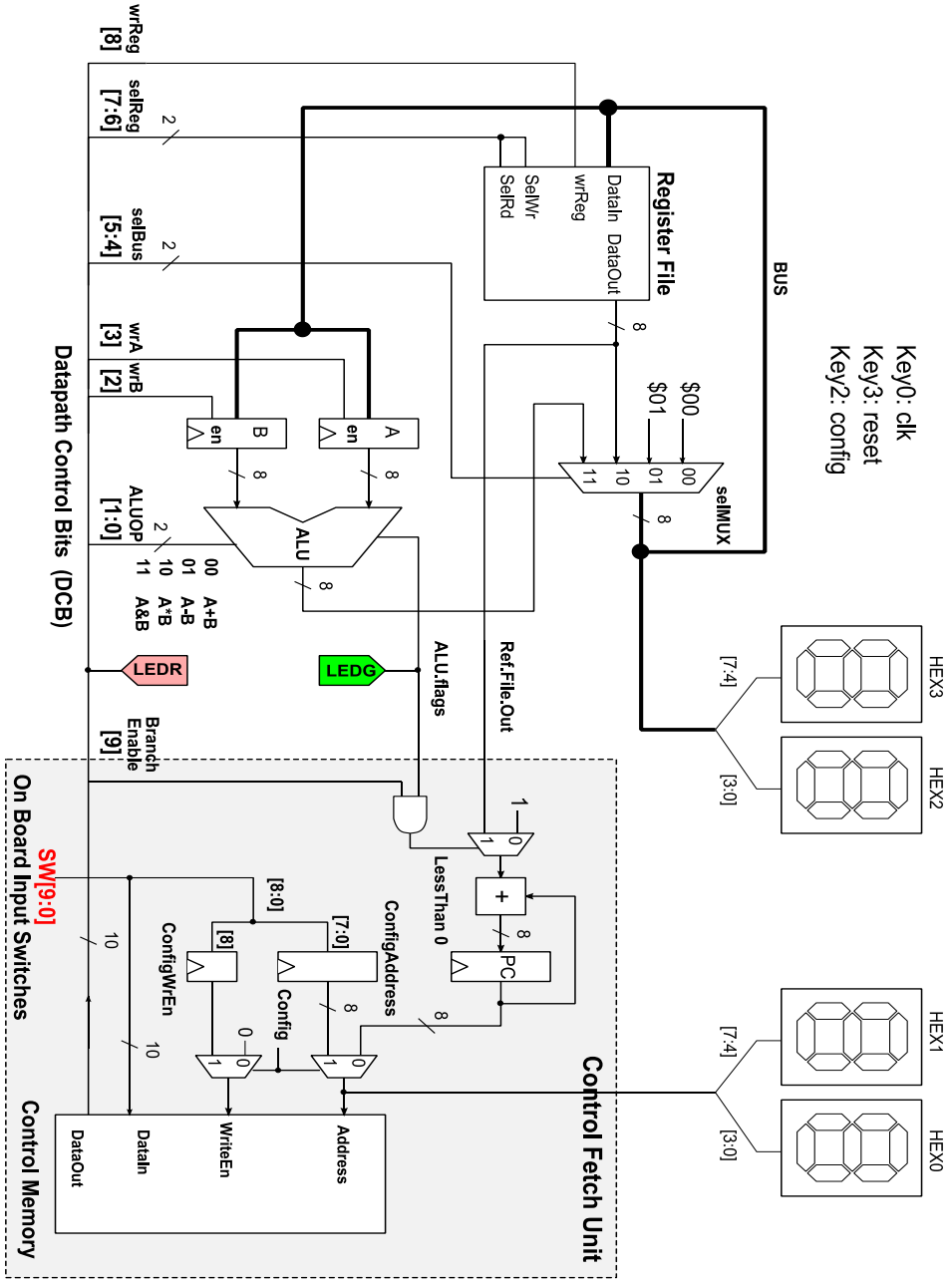
Εργαστηριακή άσκηση 2

Σκοπός αυτής της εργαστηριακής άσκησης είναι να σας θυμίσει (ή να σας δείξει ανάλογα με το βαθμό εξοικίωσης σας) τον τρόπο που μπορούμε να επαυξήσουμε τη λειτουργία του βασικού datapath ώστε να μοιάζει σε ένα κανονικό επεξεργαστή. Ο μικρός επεξεργαστής θα ολοκληρωθεί στο επόμενο εργαστήριο. Το επαυξημένο κύκλωμα το οποίο έχετε να αντιμετωπίσετε φαίνεται στο σχήμα που ακολουθεί και όπως και στο προηγούμενο εργαστήριο έχει αντιστοιχιθεί στη λογική της FPGA, ενώ τα σήματα ελέγχου (SW[9:0] και Key[3:0]) έχουν ήδη συνδεθεί με τους διακόπτες της πλακέτας του εργαστηρίου.



Στο βασικό datapath το οποίο απεικονίζεται στην πάνω πλευρά της παραπάνω εικόνας, έχει προστεθεί μια μονάδα ανάκτησης των σημάτων ελέγχου (Control Fetch Unit). Η μονάδα αυτή έρχεται να λύσει το

πρόβλημα εισόδου των σημάτων ελέγχου που παρουσίαζε το βασικό datapath: αν σε μια αλληλουχία από δυαδικά ψηφία ελέγχου συνέβαινε ένα λάθος, τότε έπρεπε όλα τα σήματα ελέγχου να ξαναπεραστούν από την αρχή.

Τα σήματα ελέγχου του datapath όπως το ξέραμε μέχρι σήμερα μπορούν να καθορίσουν τις παρακάτω λειτουργίες: Αρχικά μέσω συνδυαστικής λογικής και ξεκινώντας από τους καταχωρητές A και B υπολογίζεται η έξοδος της ALU και αμέσως μετά(στον ίδο κύκλο ρολογιού) η νέα τιμή του Bus

$$us \leftarrow \begin{cases} 0 & when\ selBus = 00 \\ 1 & when\ selBus = 01 \\ RF[selReg] & when\ selBus = 10 \\ ALUout\ (A + \quad \text{ή } A - B \text{ ή } A * B \text{ ή } A\&B) & when\ selBus = 11 \end{cases}$$

Η τιμή του Bus μεταφέρεται με την επόμενη ακμή του ρολογιού ανάλογα με τις τιμές των άλλων σημάτων ελέγχου στην έξοδο των καταχωρητών A, B ή και του αρχείου καταχωρητών (όλες μαζί ταυτόχρονα, ένα μέρος αυτών ή και καμμιά από αυτές)

$$\boldsymbol{A} \leftarrow \boldsymbol{Bus}\ if\ wrA = 1, \boldsymbol{B} \leftarrow \boldsymbol{Bus}\ if\ wrB = 1, \boldsymbol{RF}[\boldsymbol{selReg}] \leftarrow \boldsymbol{Bus}\ if\ wrReg = 1$$

Οι παραπάνω συμβολική κωδικοποίηση των λειτουργιών του datapath αποτελεί για μας το λογισμικό που μπορεί να εκτελέσει το datapath αυτό. Για παράδειγμα, αν οι τιμές των διακοπτών SW[8:0] ήταν 1 01 11 11 00 και ακολουθούσε μια ακμή του ρολογιού, αυτό θα σήμαινε την εκτέλεση των παρακάτω λειτουργίων (λειτουργίες που φαίνονται στην ίδια γραμμή εκτελούνται ταυτόχρονα ενώ αυτές που εμφανίζονται σε χωριστές γραμμές εκτελούνται η μια μετά την άλλη) :

ALUout = Aout + Bout

Bus = ALUout

Ain =Bus, Bin =Bus, RegFileDataIn=Bus, RegFileSelRD=RegFileSelWR=selReg, RegFileWrReg=WrReg

Και στη θετική ακμή του ρολογιού

Aout =Ain, Bout=Bin και RF[selReg] = RegFileDataIn.

Πιο σύντομα θα μπορούσαμε να είχαμε γράψει πως

Anew <= A+B, Bnew <= A+ B, Reg[selReg]<=A+B.

Η εκτέλεση ενός ολόκληρου προγράμματος δεν ολοκληρώνεται με μια μόνο λέξη ελέγχου αλλά απαιτεί μια αλληλουχία από αυτές. Για παράδειγμα η παρακάτω ακολουθία εκτελεί την ακολουθία Fibonacci του πρώτου εργαστηρίου για 3 επαναλήψεις και σώζει το αποτέλεσμα στον καταχωρητή R[3].

Κύκλος	**Σήματα ελέγχου**	**Συμβολική Περιγραφή**
1	0 00 01 11 00	A=1, B=1
2	0 00 11 01 00	A=1, B=2
3	0 00 11 10 00	A=3, B=2
4	0 00 11 01 00	A=3, B=5
5	1 11 11 00 00	R[3]=ALUout=8

Αν αποθηκεύσουμε την ακολουθία από σήματα ελέγχου σε μια μνήμη τότε θα μπορούσαμε διαβάζοντας κύκλο-προς-κύκλο τα περιεχόμενα της μνήμης αυτής να καθορίσουμε τη λειτουργία του datapath. Για να το κάνουμε αυτό, το κύκλωμα μας χρειάζεται έναν επιπλέον καταχωρητή ο οποίος θα «δείχνει» στην επόμενη προς εκτέλεση λέξη των σημάτων ελέγχου. Αρχικά ο καταχωρητής αυτός θα έδειχνε στη θέση 0 της μνήμης, ενώ σε κάθε κυκλο ρολογιού θα αύξανε την τιμή του κατά 1 επιτρέποντας μας να διαβάσουμε όλα τα περιεχόμενα της μνήμης των σημάτων ελέγχου. Ο καταχωρητής αυτός ονομάζεται συνήθως μετρητής προγράμματος (program counter - PC) και σύμφωνα με το σχήμα που σας δόθηκε στην αρχή, όταν config=0, οδηγεί τη γραμμή διεύθυνσης (address) της μνήμης των σημάτων ελέγχου.

Ετσι σε κάθε κύκλο τα σήματα ελέγχου που οδηγούν το datapath (DCB) δίνονται από τα περιεχόμενα της μνήμης ελέγχου στη διεύθυνση που δείχνει ο PC. Συμβολικά DCB = ControlMem[PC]. Στον ίδιο κύκλο ρολογιού πραγματοποιείται η εκτέλεση ενώ με την επόμενη ακμή του ρολογιού πραγματοποιείται η εγγραφή στους καταχωρητές A και B, στο Register File ενώ αυξάνεται και η τιμή του PC, δλδ PC = PC+1.

Για να μπορέσει το datapath αυτό να πλησιάσει όσο πιο κοντά γίνεται στη λειτουργία ενός κανονικού επεξεργαστή θα πρέπει να μπορεί να προσαρμόσει τη λειτουργία του ανάλογα με τις τιμές των δεδομένων που υπολογίζει επιτρέποντας έτσι την εκτέλεση συγκεκριμένων λειτουργιών υπο-συνθήκη “if-then-else” αλλά και τον καθορισμό επαναληπτικών εκτελέσεων “while (condition) do”. Για να πραγματοποιηθούν τέτοιου είδους λειτουργίες πρέπει ο επεξεργαστής μας να μπορεί να αλλάζει δυναμικά τη τιμή του PC δηλαδή να μπορεί να «επιλέγει» την επόμενη προς εκτέλεση αλληλουχία σημάτων ελέγχου ανάλογα με τις τιμές που γέννησαν προηγούμενα σήματα ελέγχου.

Για το λόγο αυτό η τιμή του PC μπορεί υπο-συνθήκη να αλλάξει ως εξής:

PC=PC+1 when ALUout > 0 else PC=PC+RF[selReg].

Με άλλα λόγια αν το αποτέλεσμα της ALU είναι μεγαλύτερο του μηδενός ο PC αυξάνει την τιμή του κατά 1 δείχνοντας στην επόμενη προς εκτέλεση λέξη σημάτων ελέγχου (DCB). Αν όμως η ALU υπολογίσει ένα αποτέλεσμα που είναι μικρότερο του μηδενός (δε μας νοιάζει ποια πράξη προκάλεσε το αποτέλεσμα αυτό) τότε ο PC φορτώνεται στην ακμή του ρολογιού με το άθροισμα της τρέχουσας τιμής του και της τιμής που διαβάζεται εκείνη τη στιγμή από το αρχείο καταχωρητών. Επομένως, αν θέλαμε ο PC να μεταβεί σε μια διεύθυνση που βρίσκεται 4 θέσεις πιο κάτω από το σημείο εκτέλεσης (εκεί που τώρα δείχνει ο PC) θα έπρεπε να είχαμε φροντίσει να σχηματίσουμε νωρίτερα σε έναν επιλεγμένο καταχωρητή του αρχείου καταχωρητών τη σταθερά 4.

Η επιλογή της αλλαγής της τιμής του PC δε θέλουμε να συμβαίνει σε κάθε κύκλο ρολογιού αλλά μόνο σε εκείνο τον κύκλο που το επιθυμούμε. Για το λόγο αυτό προσθέσαμε σε σχέση με την προηγούμενη άσκηση ένα επιπλέον σήμα ελέγχου το bit 9 (στο σχήμα εμφανίζεται ως branch enable [9]) το οποίο όταν είναι ενεργοποιημένο επιτρέπει την αλλαγή της τιμής του PC προς μιας άλλη διεύθυνση, ενώ όταν είναι 0 τα άλματα απαγορεύονται και ο PC θα αυξήσει απλά την τιμή του κατά ένα.

Για παράδειγμα ο παρακάτω κώδικας θα μπορούσε να εκτελεστεί απο την αλληλουχία σημάτων ελέγχου που φαίνεται στον πίνακα που ακολουθεί.

```
// variable a store in register A
// variable b stored in register B
// variable c stored in register R[2]
01: char a, b, c;
02: a = 1;
```

```
03: b = 1;
04: c = 1;
05: b = a + b;
06: a = a + b;
07: if (a > b) {
08:   c = 0;
09: }
10: a = a + c;
```

Διευθυνση	Σήματα ελέγχου	Συμβολική Περιγραφή
0	**0** 1 10 01 11 00	ALUop:A+B, A=1, B=1, R[2]=1, PC=PC+1
1	**0** 1 11 11 01 00	ALUop:A+B, A=1, B=2, R[3]=2, PC=PC+1
2	**0** 0 00 11 10 00	ALUop:A+B, A=3, B=2, PC=PC+1
3	**1** 0 11 11 00 01	ALUop:A-B, PC=PC+1 if A-B > 0 else PC=PC+R[3] (2)
4	**0** 1 10 00 00 00	R[2]=0, PC=PC+1
5	**0** 0 10 10 01 00	B=R[2], PC=PC+1
6	**0** 0 10 11 10 00	A = A + B, PC=PC+1

Το σήμα branch enable βρίσκεται στην πιο αριστερή θέση (bit [9]) και όταν ενεργοποιηθεί επιτρέπει στον PC αντί να αυξηθεί κατά 1 να αυξηθεί ανάλογα με τα περιεχόμενα του RF με δεδομένο ότι εκείνη τη στιγμή η έξοδος της ALU έχει υπολογίσει έναν αριθμό ο οποίος είναι μικρότερος ή ίσος του μηδενός. Στις άλλες περιπτώσεις που δε χρειάζεται να ελέγξουμε την ικανοποίηση ενός άλματος απλά θέτουμε το branch enable στην τιμή 0. Στο πάραδειγμα μας στην θέση της μνήμςη ελέγχου η συνθήκη της διακλάδωσης ελέγχεται στη θέση μνήμης 3. Αν η συνθήκη είναι αληθής η συνθήκη μεταβαίνει στη θέση μνήμης 4 (θέτοντας τη μεταβλητη C στην τιμή 0) αλλιώς η εκτέλεση του προγράμματος συνεχίζεται στη στη θέση μνήμης 5 (τα σήματα ελέγχου στις θέσεις 5 και 6 υλοποιούν την εντολή στη γραμμή 10 σε C).

Φόρτωση ακολουθίας σημάτων ελέγχου στη μνήμη ελέγχου

Πριν την εκτέλεση του αλγορίθμου σας στο datapath μπορείτε να φορτώσετε στη μνήμη ελέγχου ολόκληρη την ακολουθία ελέγχου (μέχρι 256 πράξεις του datapath) φέρνοντας το σύστημα σας σε configuration mode. Αυτό πραγματοποιείται μέσω του πλήκτρου Key2. Πατώντας το key2 μια φορά το σύστημα σας επιτρέπει να γεμίσετε μία-μία τις θέσεις τη μνήμη ελέγχου. Κάθε βήμα γεμίσματος της μνήμης ελέγχου αποτελείται από δύο υπο-βήματα. Στο πρώτο υπο-βήμα δίνουμε στη μνήμη ελέγχου τη διεύθυνση στην οποία θέλουμε να γράψουμε μαζί με ένα σήμα Wen (switches [7:0] και switch [8]) και ολοκληρώνουμε με ένα παλμό του ρολογιού. Μετά χρησιμοποιώντας τους ίδιους διακόπτες (switches [9:0]) καθορίζουμε τα περιεχόμενα της μνήμης ελέγχου (σήματα ελέγχου του datapath) ενώ αυτά γράφονται δίνοντας εκ νέου έν παλμό του ρολογιού. Για παράδειγμα αν ήθελα να γράψω στη μνήμη ελέγχου τα παρακάτω στοιχεία

Διεύθυνση 002: 0 0 10 11 00 01

Διεύθυνση 003: 0 0 10 11 00 00

Τότε πατώ το key2 σχηματίζω τη διεύθυνση 2 και το Wen 1 στους 9 πρώτους διακόπτες X1 0000 0010 (ο SW[9] μας είναι αδιάφορος) και δίνω clock αμέσως μετά. Στη συνέχεια αφού έχω περάσει τη διεύθυνση σχηματίζω τα περιεχόμενα της διεύθυνσης αυτής χρησιμοποιώντας και τους 10 διακόπτες (00 1011 0001). Ολοκληρώνω την εγγραφή με ένα νέο παλμό του ρολογιού. Με τον ίδιο τόπο σχηματίζω πρώτα τη διεύθυνση 3 στους διακόπτες, δλδ X1 0000 0011 και δίνω ένα παλμό του ρολογιού. Αμέσχως μετά, αφού σχηματίσω τα δεδομένα της διεύθυνσης 3 (σήματα ελέγχου του datapath) 00 1011 0000 επειδή είναι τα

τελευταία δεδομένα που θέλω να βάλω πατάω πρώτα το κουμπί Key2 για να βγώ από το configuration mode και μετά δίνω ένα παλμό του ρολογιού.

Τα LEDG[7:6] δείχνουν σε ποια κατάσταση βρίσκεται το σύστημα σας. Οταν είναι “01” σημαίνει πως βρισκόμαστε σε configuration mode και είμαστε στη φάση εισαγωγής διεύθυνσης της μνήμης ελέγχου. Αντίστοιχα όταν είναι “10” σημαίνει πως είμαστε στη φάση εισαγωγής δεδομένων στη μνήμη ελέγχου. Τέλος όταν τα leds είναι και τα δύο αναμένα “11” σημαίνει ότι έχουμε φύγει από το configuration mode και είμαστε στη φάση εκτέλεσης των προγραμμάτων όπου η μνήμη ελέγχου διευθυνσιοδοτείται πια από τον PC.

ΖΗΤΟΥΜΕΝΑ

Αφού έχετε κατανοήσει και πειραματιστεί με τη λειτουργία του επαυξημένου datapath καλείστε να υλοποιήσετε δύο λειτουργίες υψηλότερου επιπέδου πανω σε αυτή την απλή μηχανή.

Η πρώτη λειτουργία swap RA, RB αντιγράφει τα περιεχόμενα του καταχωρητή RB στον καταχωρητή RA και τα περιεχόμενα του RA στον RB.

Η δεύτερη λειτουργία jump offset μεταβιβάζει την εκτέλεση του προγράμματος στη θέση PC=PC+offset χωρίς να περιμένει την ικανοποίηση ή όχι κάποιας συνθήκης. Θεωρήστε πως η τιμή του offset θα είναι καταχωρημένη στον καταχωρητη R[3].

Η τρίτη λειτουργία είναι κάπως πιο σύνθετη και ορίζεται ως εξής: subleq RA, RB, offset.

```
RA=RA-RB
if (RA <= 0)
  PC=PC+offset
else
  PC=PC+1
```

Και πάλι η τιμή του offset βρίσκεται στον καταχωρητή R[3]. Τα RA και RB της εντολής δε μπορεί να είναι ο R[3].

Σε όλες τις περιπτώσεις μπορείτε φυσικά να χρησιμοποιήσετε και επιπλέον καταχωρητές είτε του register file είτε τους A και B ώστε να δημιουργήσετε τις τιμές που επιθυμείτε ή να κρατήσετε ενδιάμεσα αποτελέσματα. Αφού υλοποιήσετε τις παραπάνω λειτουργίες και τις τοποθετήσετε στη μνήμη ελέγχου δημιουργήστε ένα πρόγραμμα που περνά τιμές σε επιπλεγμένους τους καταχωρητές και δείχνει το αποτέλεσμα εκτέλεσης των τριών πιο πάνω λειτουργιών.