ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

10

ΜΑΘΗΜΑ: ΣΥΓΧΡΟΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΠΣΤΩΝ – ΕΡΓΑΣΤΗΡΙΟ

Ονοματεπώνυμο: ΜΗΛΤΙΑΔΗΣ ΜΑΝΤΕΣ Α.Μ.: 1084661 E-mail: up1084661@upnet.gr
Ονοματεπώνυμο: ΦΙΛΙΠΠΟΣ-ΠΑΡΑΣΚΕΥΑΣ ΖΥΓΟΥΡΗΣ Α.Μ.: 1084660 E-mail: up1084660@upnet.gr

ΑΝΑΦΟΡΑ ΑΣΚΗΣΗΣ 1

Accumulator: 1000
Program Counter: 1001
Βοηθητικός Καταχωρητής X: 1010

Ζητούμενα

Α. Να αλλάζετε την υλοποίηση των μικροπρογραμμάτων χρησιμοποιώντας τους καταχωρητές που θα σας δοθούν στο εργαστήριο. Επίσης σε κάθε μικροπρόγραμμα προσπαθήστε να μειώσετε τον αριθμό των απαιτούμενων μικροεντολών (μία μικροεντολή σε κάθε μικροπρόγραμμα μπορεί να αφαιρεθεί). Είναι απαραίτητη η χρήση του βοηθητικού καταχωρητή X;

- Ο X είναι ένας βοηθητικός καταχωρητής, του οποίου η χρήση στο μικροπρόγραμμα των εντολών LDA, ADD, STA δεν είναι απαραίτητη. Μπορούμε να πραγματοποιήσουμε standard διευθυνσιοδότηση μνήμης με το περιεχόμενο του MDR με τον εξής τρόπο:

  MDR+0→X
  X+0→NOP,MAR ⟹ MDR+0→NOP, MAR

  MDR+0→X
  X+0→MAR ⟹ MDR+0→MAR

  Απαλείφοντας τον βοηθητικό καταχωρητή X γλιτώνουμε 1 καταχωρητή και 1 μικροεντολή σε κάθε εντολή.

Άρα, θα έχουμε πλέον τις εντολές υλοποιημένες σε συμβολική γλώσσα ως εξής :

LDA $K :

PC + 1 → PC , MAR
MDR + 0 → MAR
MDR + 0 → ACC
PC + 1 → PC, MAR
NEXT(PC)

ADD $K :

PC + 1 → PC , MAR
MDR + 0 → NOP, MAR
MDR + ACC → ACC
PC + 1 → PC, MAR
NEXT(PC)

STA $K :

PC + 1 → PC , MAR
MDR + 0 → NOP,MAR
ACC + 0 → NOP, MWE~
PC + 1 → PC, MAR
NEXT(PC)

Β. Να γράψετε πρόγραμμα σε επίπεδο γλώσσας μηχανής που να προσθέτει το περιεχόμενο των θέσεων μνήμης με διευθύνσεις Α και Β και να αποθηκεύει το αποτέλεσμα στη θέση μνήμης Γ. Οι διευθύνσεις Α, Β και Γ θα σας δοθούν στο εργαστήριο.

➢ Εντελώς αυθαίρετα, ορίζουμε ένα μοναδικό opcode, διεύθυνση μικρομνήμης και έντελο για κάθε μία από τις μακροεντολές LDA $K, ADD $K, STA $K :

Mapper		
Κώδικας εντολής	Opcode/Θέση	Περιεχόμενα
LDA $K	00000001	00010000
ADD $K	00000010	00010001
STA $K	00000011	00010010

Opcodes Micromemory addresses

Main Memory		
Κώδικας εντολής	Θέση	Περιεχόμενο
LDA $08	00000000	00000001
	00000001	00001000
ADD $09	00000010	00000010
	00000011	00001001
STA $0a	00000100	00000011
	00000101	00001010

Opcode LDA
Έντελο 8
Opcode ADD
Έντελο 9
Opcode STA
Έντελο 10

Πρόγραμμα σε συμβολική γλώσσα

//Bootstrap :
Switches + 0 → PC, MAR
NEXT(PC)

//LDA $A :

PC + 1 → PC , MAR
MDR + 0 → MAR
MDR + 0 → ACC
PC + 1 → PC, MAR
NEXT(PC)

//ADD $B :

PC + 1 → PC , MAR
MDR + 0 → NOP, MAR
MDR + ACC → ACC
PC + 1 → PC, MAR
NEXT(PC)

//STA $Γ :

PC + 1 → PC , MAR
MDR + 0 → NOP,MAR
ACC + 0 → NOP, MWE~
PC + 1 → PC, MAR
NEXT(PC)

Πρόγραμμα σε γλώσσα μηχανής

BOOTSTRAP	BRA	BIN	CON	I	I	I	APORT	BPORT	DDATA	SH~	SELB	MWE~	MARCLK	MSTATUS	LDS~	PCE~	CARRYE~	MDE~	DDATAE~	ADDRESS
	(4:0)	(2:0)	(2:0)	(2:0)	(5:3)	(8:6)	(3:0)	(3:0)	(1:0)											
SW+0->PC,MAR	XXXXX	000	XXX	111	000	011	XXXX	1001	XX	X	1	1	1	0	1	0	1	1	1	m00
NEXT(PC)	XXXXX	000	XXX	XXX	XXX	001	XXXX	XXXX	XX	X	1	1	0	0	0	1	1	1	1	m01

LDA $K	BRA	BIN	CON	I	I	I	APORT	BPORT	DDATA	SH~	SELB	MWE~	MARCLK	MSTATUS	LDS~	PCE~	CARRYE~	MDE~	DDATAE~	ADDRESS
	(4:0)	(2:0)	(2:0)	(2:0)	(5:3)	(8:6)	(3:0)	(3:0)	(1:0)											
PC+1->PC,MAR	XXXXX	000	XXX	101	000	011	1001	1001	01	1	1	1	1	0	1	1	1	1	0	m02
MDR+0->ACC	XXXXX	000	XXX	111	000	011	XXXX	1000	XX	1	1	1	0	0	1	1	1	0	1	m03
ACC+0->NOP,MAR	XXXXX	000	XXX	100	000	001	1000	XXXX	XX	1	1	1	1	0	1	1	1	1	1	m04
MDR+0->ACC	XXXXX	000	XXX	111	000	011	XXXX	1000	XX	1	1	1	0	0	1	1	1	0	1	m05
PC+1->PC,MAR	XXXXX	000	XXX	101	000	011	1001	1001	01	1	1	1	1	0	1	1	1	1	0	m06
NEXT(PC)	XXXXX	000	XXX	XXX	XXX	001	XXXX	XXXX	XX	X	1	1	0	0	0	1	1	1	1	m07

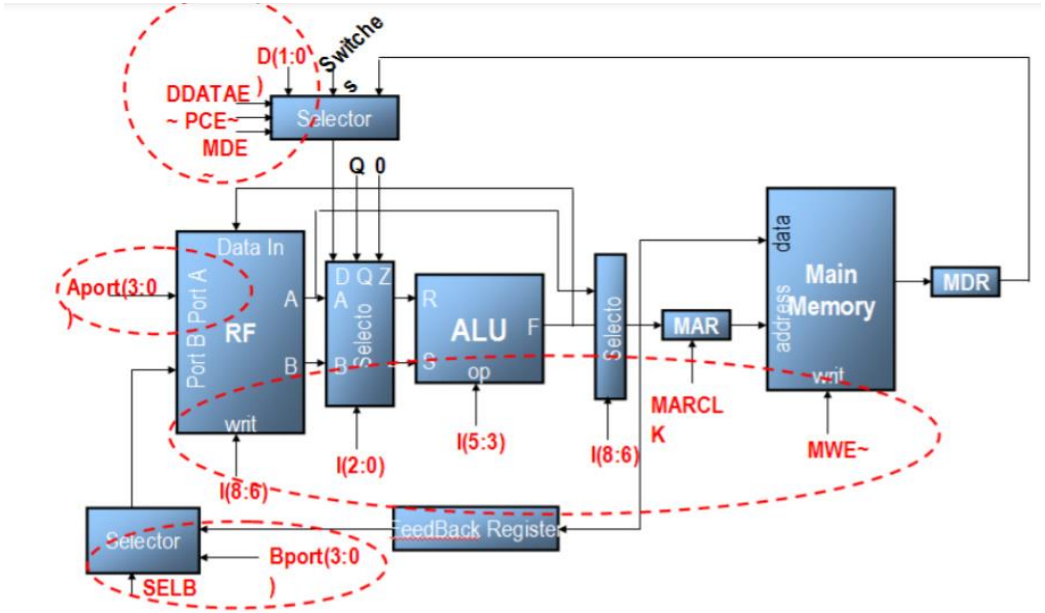
ADD $K	BRA	BIN	CON	I	I	I	APORT	BPORT	DDATA	SH~	SELB	MWE~	MARCLK	MSTATUS	LDS~	PCE~	CARRYE~	MDE~	DDATAE~	ADDRESS
	(4:0)	(2:0)	(2:0)	(2:0)	(5:3)	(8:6)	(3:0)	(3:0)	(1:0)											
PC+1->PC,MAR	XXXXX	000	XXX	101	000	011	1001	1001	01	1	1	1	1	0	1	1	1	1	0	m08
MDR+0->X	XXXXX	000	XXX	111	000	011	XXXX	1010	XX	1	1	1	0	0	1	1	1	0	1	m09
X+0->NOP,MAR	XXXXX	000	XXX	100	000	001	0010	XXXX	XX	1	1	1	1	0	1	1	1	1	1	m0a
MDR+ACC->ACC	XXXXX	000	XXX	111	000	011	XXXX	1000	XX	1	1	1	0	0	1	1	1	0	1	m0b
PC+1->PC,MAR	XXXXX	000	XXX	101	000	011	1001	1001	01	1	1	1	1	0	1	1	1	1	0	m0c
NEXT(PC)	XXXXX	000	XXX	XXX	XXX	001	XXXX	XXXX	XX	X	1	1	0	0	0	1	1	1	1	m0d

STA $K	BRA	BIN	CON	I	I	I	APORT	BPORT	DDATA	SH~	SELB	MWE~	MARCLK	MSTATUS	LDS~	PCE~	CARRYE~	MDE~	DDATAE~	ADDRESS
	(4:0)	(2:0)	(2:0)	(2:0)	(5:3)	(8:6)	(3:0)	(3:0)	(1:0)											
PC+1->PC,MAR	XXXXX	000	XXX	101	000	011	1001	1001	01	1	1	1	1	0	1	1	1	1	0	m0e
MDR+0->X	XXXXX	000	XXX	111	000	011	XXXX	1010	XX	1	1	1	0	0	1	1	1	0	1	m0f
X+0->NOP,MAR	XXXXX	000	XXX	100	000	001	1010	XXXX	XX	1	1	1	1	0	1	1	1	1	1	m10
ACC+0->NOP,MWE	XXXXX	000	XXX	100	000	001	1000	XXXX	XX	1	1	1	0	0	1	1	1	1	1	m11
PC+1->PC,MAR	XXXXX	000	XXX	101	000	011	1001	1001	01	1	1	1	1	0	1	1	1	1	0	m12
NEXT(PC)	XXXXX	000	XXX	XXX	XXX	001	XXXX	XXXX	XX	X	1	1	0	0	0	1	1	1	1	m13

Χρήση Don't care bits:

1. Μειώνουν την αλλαγή των καταστάσεων κάτι το οποίο συνεπάγεται με την μείωση του χώρου μνήμης που απαιτείται για την αναπαράσταση ενός δεδομένου ψηφιακού κυκλώματος που με τη σειρά του έχει ως αποτέλεσμα λιγότερη κατανάλωση ενέργειας.
2. Απλοποίηση εξόδου κυκλώματος.
3. Αποτρέπουν κινδύνους.
4. Διευκολύνουν σε μετατροπές κώδικα.

Έστω διευθύνσεις A: XXXX, B: YYYY, τις οποίες βάζουμε στο Port A και Port B αντίστοιχα. Ενεργοποιούμε το σήμα write=1, οπότε τα περιεχόμενα της διεύθυνσης A (XXXX) βγαίνουν από την έξοδο A. Η έξοδος A ή B θα μου δώσει τα περιεχόμενα του καταχωρητή που διευθυνσιοδοτήσαμε στο Port A ή B αντίστοιχα. Τα τροφοδοτούμε στην είσοδο της ALU, αφού θέλουμε να κάνουμε πρόσθεση βάζουμε op=000, οπότε θα πάρει αυτό που έχει σαν είσοδο από το R (έξοδος A του register file), θα το προσθέσει με αυτό που έχει σαν είσοδο από το S (έξοδος B του register file) και το αποτέλεσμα θα το βάλει στην έξοδο F. Δεν πειράζουμε την μνήμη, κάνοντας enable το MAR ώστε να κλειδώσουμε μια τιμή και στο MWE~=0 (σήμα write της κύριας μνήμης). Το αποτέλεσμα της πρόσθεσης θα αποθηκευτεί στην διεύθυνση του καταχωρητή στο PortB, το οποίο είναι το μόνο port εγγραφής.



Γ. Εάν είχατε στη διάθεσή σας μόνο τις εντολές που υλοποιήσατε, από πόσες εντολές θα αποτελούνταν ένα πρόγραμμα που θα έκανε τον υπολογισμό Γi = Ai+Bi για i=10; Να δικαιολογήσετε την απάντησή σας.

- Για να εκτελέσουμε 10 αθροίσεις θα χρησιμοποιήσουμε 10 φορές τις τριάδες μακροεντολών LDA – ADD – STA στη κύρια μνήμη, άρα συνολικά θα χρειαστούμε 30 μακροεντολές.

Σημείωση:

- Γενικά για μια εντολή που δέχεται K ορίσματα θα πρέπει να κάνουμε PC+1→PC,MAR K φορές μια για κάθε όρισμα. Χρειαζόμαστε ακόμα μια PC+1→PC,MAR για να διευθυνσιοδοτήσουμε το opcode της επόμενης μακροεντολής. Συνεπώς χρειαζόμαστε (K+1): PC+1→PC,MAR για κάθε εντολή ,επειδή έχουμε 3 εντολές(LDA,ADD,STA) θα χρειαστούμε συνολικά 3*(K+1) PC+1→PC,MAR
  Συνολικά θα είναι f(K)=10*(3*(K+1) + Υπόλοιπες εντολές της LDA + Υπόλοιπες εντολές της ADD + Υπόλοιπες εντολές της STA),όπου:
  Υπόλοιπες εντολές i={LDA,ADD,STA} ,εξαρτώνται και αυτές από το K.

Συνεπώς. το πλήθος των μακροεντολών αποτελεί μια συνάρτηση η οποία μεταβάλλεται γραμμικά με το Κ, δηλαδή όσο ο αριθμός των ορισμάτων αυξάνεται τόσο και ο αριθμός των μακροεντολών αυξάνεται και έτσι παρατηρούμε το παρακάτω πρόβλημα.

Δ. Πως θα μπορούσατε να ξεπεράστε το πρόβλημα; Να αναφέρετε το κύριο μειονέκτημα της προτεινόμενης λύσης.

- Η χρήση τόσο πολλών εντολών για την εκτέλεση 10 προσθέσεων δεσμεύει μεγάλο τμήμα μνήμης και δαπανάται έτσι και αρκετός χρόνος προκειμένου να εκτελεστούν σειριακά όλες. Για να αντιμετωπίσουμε αυτό το θέμα θα μπορούσαμε να χρησιμοποιήσουμε εντολές διακλάδωσης (branch) και κλήσης υπορουτίνας (subroutine), έτσι ώστε οι τιμές Ai, Bi να υπάρχουν αποθηκευμένες σε 2 πίνακες και τα αποτελέσματα των αθροισμάτων Γi να αποθηκεύονται σε ένα τρίτο πίνακα. Σε κάθε εκτέλεση της υπορουτίνας θα φορτώνεται η τιμή Ai στον Accumulator, θα προστίθεται η τιμή Bi σε αυτόν και το αποτέλεσμα θα αποθηκεύεται από τον Accumulator στη θέση μνήμης Γi. Επομένως, θα χρειαστεί στις μικροεντολές που θα χρησιμοποιήσουμε να δώσουμε τιμές και στα control signals της Μονάδας Ελέγχου.