Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πολυτεχνική Σχολή Πανεπιστημίου Πατρών

Ασκήσεις Εργαστηρίου Αρχιτεκτονικής Υπολογιστών

Δρ. Γεώργιος Κεραμίδας,
ΠΔ407

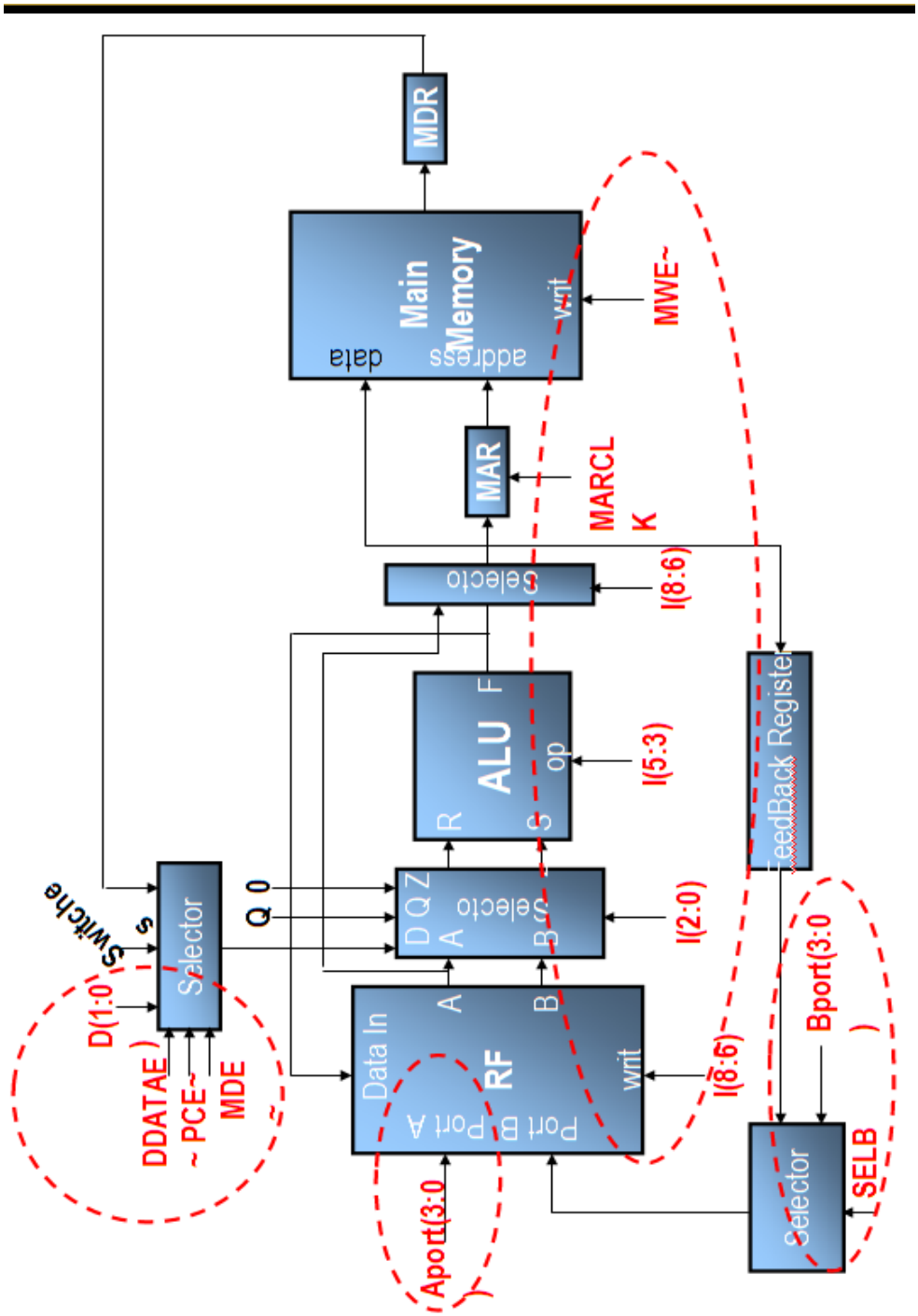
Δρ. Δημήτριος Νικολός,
Καθηγητής

Πάτρα
Φεβρουάριος 2022

Συνοπτική Εξήγηση Βασικών Σημάτων

- **Aport(3:0)** και **Bport(3:0)**: για διευθυνσιοδότηση του Register File
- **SELB:**
  - **SELB=1**, τότε το Port B διευθυνσιοδοτείται με την τιμή του **Bport(3:0)**
  - **SELB=0**, τότε το Port B διευθυνσιοδοτείται με την τιμή του **Feedback Register**, ΔΗΛΑΔΗ με το αποτέλεσμα της προηγούμενης πράξη
- **DDATAE~ = 0** , τότε απ' τον selector περνάει μία σταθερά των 2bit **D(1:0)**
- **PCE~ = 0** , τότε απ' τον selector περνάει η τιμή που έχουν 8 **DIP Switches** της πλακέτας
- **MDE~ = 0** , τότε απ' τον selector περνάει η τιμή του **MDR**
- **I(2:0)** : Επιλέγει 2 από τις 5 εισόδους του selector και τροφοδοτεί τις εισόδους R και S της ALU. Συνηθισμένες τιμές : AB:001, ZA:100, DA:101, DZ:111
- **I(5:3)** : 3 bit σήμα που καθορίζει την πράξη που θα εκτελέσει η ALU (8 επιλογές)
  - Πρόσθεση I(5:3)=000
- **I(8:6)**: 3 bit σήμα που καθορίζει: (α.) Τι θα περάσει στο σημείο **Y,** (β.) Αν θα γίνει εγγραφή στο Register File, (γ.) Αν θα γίνει εγγραφή στον Q register, (δ.) Αν θα εφαρμοστεί shift πριν την αποθήκευση
  - **NOP (001)** : Καμία εγγραφή, κανένα shift , ο selector επιλέγει το F για την έξοδο Y
  - **RAMF (011)** : Γράψε το F στο register file, όχι εγγραφή στον Q, κανένα shift, το F περνάει στο Y
- **MARCLK** : Καθορίζει αν το Y θα 'κλειδωθεί' στον MAR
- **MWE~** : Καθορίζει αν θα γίνει εγγραφή στην Κύρια Μνήμη
- **SH~:** Σε συνδυασμό με το **I(8:6)**, καθορίζει αν θα γίνει απλή ή κυκλική ολίσθηση στα αποτελέσματα (Πίνακας 1.9 σελ. 21, Σχήματα 1.7 και 1.8 σελ. 19)
- **CARRYE~ (carry enable):** Καθορίζει αν η ALU θα εκτελέσει πράξη με κρατούμενο εισόδου (Πίνακας 1.8 – σελ. 20)
- **MSTATUSCLK :** οι τιμές των micro-flags 'κλειδώνονται' στα macro-flags.
- **BIN(2:0) :** (σελ.34. Πίνακας 2.2) Καθορίζει αν η μικροεντολή είναι: α. Εντολή διακλάδωσης χωρίς συνθήκη (JMP - Jump), β. Εντολή διακλάδωσης με συνθήκη (BR – Branch conditional), γ. Κλήση υπορουτίνας με ή χωρίς συνθήκη, δ. Επιστροφή από υπορουτίνα με ή χωρίς συνθήκη, ε. Σειριακή μικροεντολή (000)
- **CON(2:0) :** (σελ. 35, Πίνακας 2.3) Καθορίζει ποια είναι η συνθήκη για conditional εντολές (Π.χ Branch if macroCarry ή Branch if microCarry)
- **BRA(4:0) :** 5-bit πεδίο το οποίο προστίθεται (2's complement, εύρος [-16,15)) στη διεύθυνση της τρέχουσας μικροεντολής για την εύρεση της διεύθυνσης της επόμενης μικροεντολής (σημ: μόνο αν πρόκειται για εντολή που αλλάζει τη ροή π.χ. JMP, taken branch κλπ.)
- **LOAD_DIR_SEQ~** : Με την ενεργοποίηση του σήματος αυτού η έξοδος του mapper περνάει στον μPC (και διευθυνσιοδοτεί τη μικρομνήμη). Ενεργοποιούμε το σήμα αυτό μόνο όταν ισχύουν τα δύο παρακάτω (και τα δύο, πρακτικά είναι η τελευταία εντολή σε κάθε μικροπρόγραμμα) :
  - α. όλες οι λειτουργίες του τρέχοντος μικροπρογράμματος έχουν ολοκληρωθεί
  - β. ο MAR δείχνει στο opcode της επόμενης προς εκτέλεση εντολής
  - Μετά την ενεργοποίηση ο έλεγχος θα περάσει στο μικροπρόγραμμα που αντιστοιχεί στην επόμενη εντολή (του προγράμματος)

Γενικό Διάγραμμα της Πλακέτας



Περιληπτική Περιγραφή των Ασκήσεων

Άσκηση 1

Υλοποίηση βασικών εντολών (εισαγωγή).

Άσκηση 1: Υλοποίηση βασικών εντολών (εισαγωγή)

Ο στόχος της τρέχουσας άσκησης είναι να γραφούν μικροπρογράμματα (ομάδες μικροεντολών) για την υλοποίηση μερικών βασικών εντολών ενός επεξεργαστή που βασίζεται στη χρήση συσσωρευτή (Accumulator). Δηλαδή θα χρησιμοποιήσουμε τη μηχανή μας για εκτέλεση εντολών ενός επεξεργαστή που βασίζεται στη χρήση συσσωρευτή (εξομοίωση ενός επεξεργαστή που βασίζεται στη χρήση συσσωρευτή). Σύμφωνα με τον κλασσικό ορισμό, ο προγραμματιστής σε επίπεδο εντολών γλώσσας μηχανής βλέπει μόνο ένα καταχωρητή, τον οποίο καλούμε συσσωρευτή. Κάθε πράξη γίνεται μεταξύ του περιεχομένου του συσσωρευτή και μιας θέσης μνήμης και το αποτέλεσμα αποθηκεύεται στον συσσωρευτή. Από τους 16 καταχωρητές να χρησιμοποιήσετε ένα ως Μετρητή Προγράμματος και ένα ως συσσωρευτή. Υπενθυμίζουμε ότι ο Μετρητής Προγράμματος (Program Counter) είναι ένας καταχωρητής ο οποίος κάθε φορά περιέχει (δείχνει) τη διεύθυνση της θέσης μνήμης που περιέχει την επόμενη προς εκτέλεση εντολή.

Ζητούμενα Μικροπρογράμματα

Να γραφούν μικροπρογράμματα για την υλοποίηση των ακόλουθων εντολών.

- **LDA $K :** φόρτωσε στο συσσωρευτή το περιεχόμενο της θέσης μνήμης με διεύθυνση K.
- **ADD $K :** πρόσθεσε, χωρίς κρατούμενο, το περιεχόμενο του συσσωρευτή με το περιεχόμενο της θέσης μνήμης με διεύθυνση K και αποθήκευσε το αποτέλεσμα στο συσσωρευτή.
- **STA $K :** αποθήκευσε το περιεχόμενο του συσσωρευτή στη θέση μνήμης με διεύθυνση K.

Ψευδοκώδικας Μικροπρογραμμάτων

Στην συνέχεια δίνεται ο ψευδοκώδικας των τριών ζητούμενων μικροπρογραμμάτων.

LDA $K : Φόρτωσε τον Accumulator με το περιεχόμενο της διεύθυνσης K (της κύριας μνήμης)

1. PC + 1 → PC , MAR // Το K θα πάει στον MDR
2. MDR + 0 → X // Το K στον accumulator
3. X + 0 → MAR // Το K στον MAR, άρα το περιεχόμενο της K στον MDR
4. MDR + 0 → ACC // Ο MDR (δηλ. το περιεχ. της K) στον ACC
5. PC + 1 → PC, MAR // Ο MAR θα δείξει στο opcode της επόμ. εντολής
6. NEXT(PC) // LOAD_DIR_SEQ~ για να πάμε στο μικροπρόγραμμα της

// επόμενης εντολής του προγράμματος

STA $K : Αποθήκευσε το περιεχόμενο του Accumulator στη θέση μνήμης με διεύθυνση K

1. PC + 1 → PC , MAR

2. MDR + 0 → **X**

3. **X** + 0 → NOP,MAR

4. ACC + 0 → NOP, MWE~

5. PC + 1 → PC, MAR

6. NEXT(PC)

ADD $K : Πρόσθεσε στον Accumulator το περιεχόμενο της διεύθυνσης K

1. PC + 1 → PC , MAR

2. MDR + 0 → **X**

3. **X** + 0 → NOP, MAR

4. MDR + ACC → ACC

5. PC + 1 → PC, MAR

6. NEXT(PC)

Όπου X είναι ένας βοηθητικός καταχωρητής.

40-αδες

Στην συνέχεια δίνονται τα περιεχόμενα της μικρομνήμης, του mapper και της κύριας μνήμης του συστήματος. Θεωρήσαμε ότι: Accumulator: 00H, Program Counter: 01H, Βοηθ. Καταχωρητής X: 02H

```
MICRO
    12345 678 901 234    567    890    1234 5678 90  1234567890
    BRA   BIN CON I(2:0) I(5:3) I(8:6) A    B    DD  Control signals
m00 00000 000 000 111    000    011    0000 0001 00  0111010111 // Bootstrap
m01 00000 000 000 000    000    001    0000 0000 00  0010000000 // Bootstrap
m02 00000 000 000 101    000    011    0001 0001 01  0111011110 // εντολή LDA
m03 00000 000 000 111    000    011    0000 0010 00  0110011101
m04 00000 000 000 100    000    001    0010 0000 00  0111011111
m05 00000 000 000 111    000    011    0000 0000 00  0110011101
m06 00000 000 000 101    000    011    0001 0001 01  0111011110
m07 00000 000 000 000    000    001    0000 0000 00  0010000000
m08 00000 000 000 101    000    011    0001 0001 01  0111011110 // εντολή
ADD
m09 00000 000 000 111    000    011    0000 0010 00  0110011101
m0a 00000 000 000 100    000    001    0010 0000 00  0111011111
m0b 00000 000 000 101    000    011    0000 0000 00  0110011101
m0c 00000 000 000 101    000    011    0001 0001 01  0111011110
m0d 00000 000 000 000    000    001    0000 0000 00  0010000000
m0e 00000 000 000 101    000    011    0001 0001 01  0111011110 // εντολή STA
m0f 00000 000 000 111    000    011    0000 0010 00  0110011101
m10 00000 000 000 100    000    001    0010 0000 00  0111011111
m11 00000 000 000 100    000    001    0000 0000 00  0100011111
m12 00000 000 000 101    000    011    0001 0001 01  0111011110
m13 00000 000 000 000    000    001    0000 0000 00  0010000000
```

```
MAPPER      // opcode εντολής ➔ αρχική διεύθυνση μικρομνήμης
m00 02      // εντολή LDA
m01 08      // εντολή ADD
m02 0e      // εντολή STA
MAIN
m00 00      // opcode εντολής LDA
m01 08      // έντελο εντολής LDA
m02 01      // opcode εντολής ADD
m03 09      // έντελο εντολής ADD
m04 02      // opcode εντολής STA
m05 0a      // έντελο εντολής STA
m06 f0
m07 ff
m08 03      // περιοχή δεδομένων
m09 02
m0a 01
```

Ζητούμενα

Α. Να αλλάξετε 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 ;

Β. Να γράψετε πρόγραμμα σε επίπεδο γλώσσα 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.

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

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