Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πολυτεχνική Σχολή Πανεπιστημίου Πατρών

Ασκήσεις Εργαστηρίου Αρχιτεκτονικής Υπολογιστών

Δρ. Γεώργιος Κεραμίδας,
ΠΔ407

Δρ. Δημήτριος Νικολός,
Καθηγητής

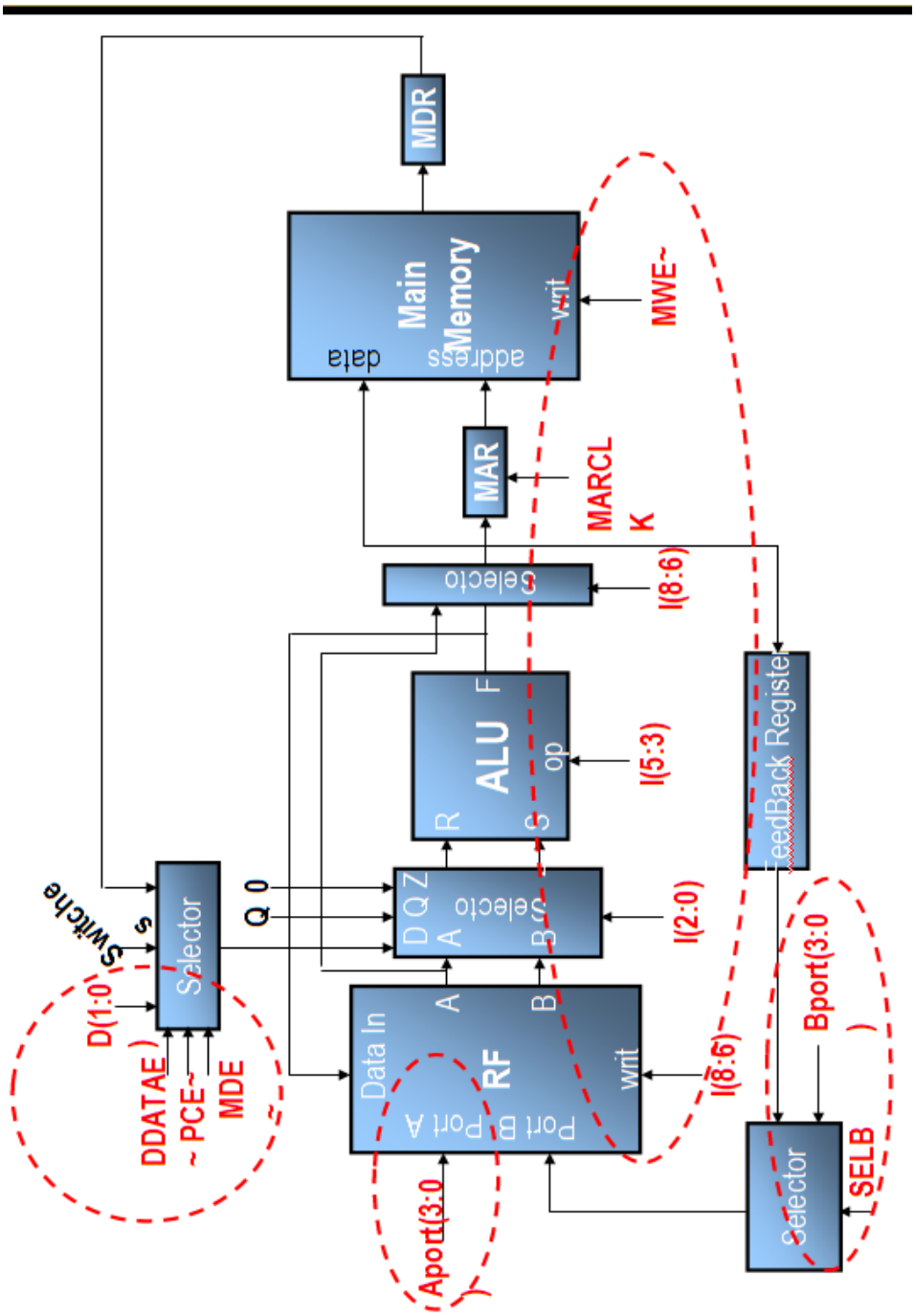
Πάτρα
Φεβρουάριος 2012

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 μακροεντολής
- Μετά την ενεργοποίηση ο έλεγχος θα περάσει στο μικροπρόγραμμα που αντιστοιχεί στην επόμενη μακροεντολή (του μακροπρογράμματος).

Γενικό Διάγραμμα της Πλακέτας



Περιληπτική Περιγραφή των Ασκήσεων

Άσκηση 1

Υλοποίηση βασικών εντολών (εισαγωγή).

Άσκηση 2

Υλοποίηση βασικών τρόπων διευθυνσιοδότησης.

Άσκηση 3

Υλοποίηση ενός απλού συνόλου εντολών μηχανής μιας διεύθυνσης.

Άσκηση 4

Υλοποίηση μηχανισμού στοίβας.

Άσκηση 5

Υλοποίηση βασικών RISC εντολών σε μηχανή δύο διευθύνσεων.

Άσκηση 6

Υλοποίηση διανυσματικών εντολών.

Άσκηση 7

Υλοποίηση βασικών εντολών του Intel 8085

Άσκηση 1: Υλοποίηση βασικών εντολών (εισαγωγή)

Ο στόχος της τρέχουσας άσκησης είναι να υλοποιηθούν μερικές βασικές μακροεντολές σε μικροκώδικα (ομάδες μικροεντολών). Προτού προχωρήσουμε στην υλοποίηση των μικροεντολών θα πρέπει πρώτα να ορίσουμε μερικούς βασικούς καταχωρητές όπως είναι ο PC (Program Counter), ο Accumulator και κάποιος επιπλέον καταχωρητής (βοηθητικός καταχωρητής). Αυτοί οι καταχωρητές επιλέγονται αυθαίρετα από το 16 θέσεων αρχείο καταχωρητών του συστήματος (register file). Θα πρέπει να τονίσουμε ότι η έννοια του accumulator χρησιμοποιείται καταχρηστικά στο σύστημα μας, αφού ως accumulator ονομάσουμε είτε τον ένα και μοναδικό αρχιτεκτονικό καταχωρητή ενός συστήματος (κλασσικός ορισμός), είτε κάποιον καταχωρητή ο οποίος έχει περισσότερες αρμοδιότητες από τους υπόλοιπους καταχωρητές του συστήματος (π.χ. ο accumulator στο Intel 8085). Σε όλες τις περιπτώσεις, θεωρούμε ότι το μήκος της κωδικού λειτουργίας (opcode) της κάθε εντολής είναι ίσο με 1 byte. Επίσης, σε κάθε εντολή, η πρώτη ψηφιολέξη χρησιμοποιείται για τον κωδικό λειτουργίας και κάθε μία από τις επόμενες ψηφιολέξεις που ακολουθούν χρησιμοποιείται για τον καθορισμό του τελούμενου. Τέλος, ο Program Counter είναι ένας καταχωρητής ειδικού σκοπού (συνήθως μη-ορατός από τον προγραμματιστή) ο οποίος κάθε φορά περιέχει (δείχνει) στην επόμενη προς εκτέλεση εντολή (μακροεντολή).

Ζητούμενες Μικροεντολές

Να γραφούν μικροπρογράμματα για την υλοποίηση των ακόλουθων μακροεντολών.

- **LDA $K :** φόρτωση στον accumulator το περιεχόμενο της θέσης μνήμης με διεύθυνση K.
- **ADD $K :** πρόσθεση χωρίς κρατούμενο στον accumulator το περιεχόμενο της θέσης μνήμης με διεύθυνση K.
- **STA $K :** αποθήκευση του περιεχομένου του accumulator 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 Κ) στον ACC

5. PC + 1 → PC, MAR // Ο MAR θα δείξει στο opcode της επόμ. μακροεντολής

6. NEXT(PC) // LOAD_DIR_SEQ~ για να πάμε στο μικροπρόγραμμα της

// επόμενης μακροεντολής του μακροπρογράμματος

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

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 Κ

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 Χ είναι ο 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 μακροεντολής ➔ 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

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 ;

Άσκηση 2: Υλοποίηση βασικών τρόπων διευθυνσιοδότησης

Ο σκοπός της συγκεκριμένης άσκησης είναι να υλοποιηθούν διάφοροι μέθοδοι διευθυνσιοδότησης. Αρχικά θα πρέπει να ορισθούν ο accumulator, ο program counter και ένας βοηθητικός καταχωρητής X.

Ζητούμενες Μικροεντολές

Να γραφούν μικροπρογράμματα για την υλοποίηση των ακόλουθων μακροεντολών.

- **LDA #K :** φόρτωση στον accumulator τον δεκαεξικό αριθμό K.
- **LDX #K :** φόρτωση στον βοηθητικό καταχωρητή X τον δεκαεξικό αριθμό K.
- **LDA ($K) :** φόρτωση στον accumulator το περιεχόμενο της θέσης μνήμης με διεύθυνση το περιεχόμενο της θέσης μνήμης με διεύθυνση K.
- **LDAX :** φόρτωση στον accumulator το περιεχόμενο της θέσης μνήμης με διεύθυνση το περιεχόμενο του καταχωρητή X.
- **LDA $K,X :** φόρτωση στον accumulator με το περιεχόμενο της θέσης μνήμης με διεύθυνση ίση με R(X)+K (άθροισμα του δεκαεξαδικού αριθμού K και του περιεχομένου του βοηθητικού καταχωρητή X).
- **LDA ($K,X) :** φόρτωση στον accumulator με το περιεχόμενο της θέσης μνήμης με διεύθυνση ίση με M(K)+R(X) (άθροισμα του περιεχομένου της θέσης της κύριας μνήμης με διεύθυνση K και του περιεχομένου του βοηθητικού καταχωρητή X).
- **STA $K :** αποθήκευση του περιεχομένου του accumulator στη θέση μνήμης με διεύθυνση K.

Ζητούμενο Μακροπρόγραμμα

Να γραφεί πρόγραμμα στο οποίο θα καλούνται όλες οι παραπάνω εντολές επιλέγοντας διάφορες τιμές για τα έντελα K. Επίσης μετά από κάθε εντολή LDA θα τοποθετείτε και μια εντολή STA, ώστε να μπορεί εύκολα να επιβεβαιώνεται η σωστή λειτουργία της αντίστοιχης LDA εντολής. Οι τιμές των καταχωρητών ACC, PC και X θα σας δοθούν κατά την διάρκεια του εργαστηρίου.

Άσκηση 3: Υλοποίηση ενός απλού συνόλου εντολών σε μηχανή μιας διεύθυνσης

Ο σκοπός της συγκεκριμένης άσκησης είναι η υλοποίηση βασικών αριθμητικών εντολών και εντολών ελέγχου ροής προγράμματος θεωρώντας ένα σύνολο εντολών μιας διεύθυνσης.

Ζητούμενες Μικροεντολές

Να γραφούν μικροπρογράμματα για την υλοποίηση των ακόλουθων μακροεντολών.

- **LDA $K,X :** φόρτωση στον accumulator με το περιεχόμενο της θέσης μνήμης με διεύθυνση ίση με R(X)+K (άθροισμα του δεκαεξαδικού αριθμού K και του περιεχομένου του βοηθητικού καταχωρητή X).
- **LDX #K :** φόρτωση στον βοηθητικό καταχωρητή X τον δεκαεξαδικό αριθμό K.
- **INX :** αύξηση κατά ένα του περιεχομένου του βοηθητικού καταχωρητή X.
- **CMPX #Y :** σύγκριση του περιεχομένου του βοηθητικού καταχωρητή X με τον δεκαεξαδικό αριθμό Y.
- **STA $K,X :** αποθήκευση του accumulator στη θέσης μνήμης με διεύθυνση R(X) + K.
- **ADC $K,X :** πρόσθεση του accumulator με το περιεχόμενο της θέσης μνήμης με διεύθυνση R(X) + K χρησιμοποιώντας κρατούμενο και αποθήκευση του αποτελέσματος στον accumulator.
- **CRC :** καθαρισμός του flag κρατουμένου (C=0).
- **BRNZ $K :** κάνε branch στη διεύθυνση K (η διεύθυνση K θα είναι η νέα τιμη του program counter) αν το zero flag είναι clear (Z=0).
- **SHLA :** αριστερή ολίσθηση στα περιεχόμενα του accumulator.
- **HALT :** τέλος εκτέλεσης του προγράμματος.

Ζητούμενο Μακροπρόγραμμα

Να γραφεί πρόγραμμα για την εκτέλεση της παρακάτω επαναληπτικής διαδικασίας:

W[i] = Y[i] + 2* Z[i], όπου i=0 έως 7. Οι προσθέσεις θα πρέπει να χρησιμοποιούν το κρατούμενο που προέκυψε από την πρόσθεση της προηγούμενης επανάληψης. Οι αρχικές τιμές των W, Y και Z θα σας δοθούν κατά την διάρκεια του εργαστηρίου.

Άσκηση 4: Υλοποίηση Μηχανισμού Στοίβας (Stack)

Να υλοποιηθεί ένας μηχανισμός στοίβας (stack) που να χρησιμοποιεί σαν στοίβα 8 θέσεις της κύριας μνήμης. Για το ορισμό της στοίβας θα πρέπει να ορισθούν τρεις καταχωρητές: ο καταχωρητής βάσης (base register), ο καταχωρητής ορίου (limit register) και ο καταχωρητής δείκτη (stack pointer). Ο stack pointer (SP) θα πρέπει να δείχνει κάθε φορά στην κορυφή της στοίβας (TOS – top of the stack), ενώ οι άλλοι δύο καταχωρητές θα πρέπει να καθορίζουν την αρχή και τέλος της στοίβας. Επίσης, ο base register θα πρέπει να δείχνει υποχρεωτικά σε υψηλότερη θέση μνήμης από τον limit register. Για περισσότερες λεπτομέρειες για τον τρόπο λειτουργίας του stack αναφερθείτε στο βιβλίο της θεωρίας.

Ζητούμενες Μικροεντολές

Να γραφούν μικροπρογράμματα για την υλοποίηση των ακόλουθων μακροεντολών.

- **LOADBR #K :** αρχικοποίηση του base register με την τιμή K.
- **LOADSP #K :** αρχικοποίηση του stack pointer με την τιμή K.
- **LOADLR #K :** αρχικοποίηση του limit register με την τιμή K.
- **PUSH $K :** τοποθέτηση στην κορυφή του σωρού τα περιεχόμενα της θέσης μνήμης με διεύθυνση K.
- **POP $K :** αποθήκευση στη θέση μνήμης με διεύθυνση K το περιεχόμενο της κορυφής της στοίβας και αφαίρεση της τιμής από της στοίβα.
- **ADD :** πρόσθεση των περιεχομένων των δύο πρώτων θέσεων της στοίβας και επιστροφή του αποτελέσματος στην κορυφή της στοίβας.
- **SUB :** αφαίρεση των περιεχομένων της δεύτερης θέσης της στοίβας από την πρώτη και αποθήκευση του αποτελέσματος στην κορυφή της στοίβας.

Υποδείξεις: Κάθε φορά που κάνουμε PUSH θα πρέπει μειώνουμε τον stack pointer κατά ένα (SP=SP-1) έτσι ώστε το καινούργιο στοιχείο να πάει στην επόμενη θέση μνήμης – νέο TOS (top of the stack) - και να τοποθετούμε το καινούριο στοιχείο στη νέα θέση που δείχνει ο SP. Επίσης, θα πρέπει να γίνεται υποχρεωτικά έλεγχος για το αν η στοίβα είναι γεμάτη. Αν είναι γεμάτη τότε η PUSH δεν θα εκτελείται (δεν θα εκτελούνται οι αντίστοιχες μικροεντολές) και θα πρέπει να συνεχίσουμε με την επόμενη προς εκτέλεση μακροεντολή. Στην περίπτωση του POP θα πρέπει να αποθηκεύουμε το περιεχόμενο της θέσης μνήμης που δείχνει ο SP (TOS) στην ζητούμενη θέση της κύριας μνήμης και στη συνέχεια να αυξάνουμε

την τιμή του SP κατά ένα (SP=SP+1) έτσι ώστε η τρέχουσα κορυφή να γίνει η προηγούμενη θέση μνήμης . Θα πρέπει επίσης να γίνεται κατάλληλος έλεγχος για το αν η στοίβα είναι άδεια. Σε αυτήν την περίπτωση η POP δεν θα πρέπει να εκτελείται και το πρόγραμμα θα πρέπει να συνεχίζει με την επόμενη προς εκτέλεση μακροεντολή. Οι ADD και SUB θα πρέπει να βγάζουν τα δύο πρώτα στοιχεία της στοίβας και να τοποθετούν το αποτέλεσμα στην κορυφή της στοίβας. Τέλος, θα πρέπει στο μικροκώδικα των ADD και SUB να υπάρχει ο κατάλληλος έλεγχος ώστε να ελέγχεται αν στη στοίβα υπάρχουν τουλάχιστον δύο στοιχεία. Αν δεν υπάρχουν τότε το πρόγραμμα θα πρέπει να συνεχίζει με την επόμενη προς εκτέλεση μακροεντολή.

Ζητούμενο Μακροπρόγραμμα

Να γραφεί πρόγραμμα για τον υπολογισμό της έκφρασης: W = X – Y + Z, όπου τα W, X, Y και Z είναι θέσεις της κύριας μνήμης. Οι τιμές των W, X, Y και Z καθώς και οι τιμές των limit και base resisters θα σας δοθούν κατά την διάρκεια του εργαστηρίου.

Άσκηση 5: Υλοποίηση βασικών RISC εντολών σε μηχανή δύο διευθύνσεων

Ο στόχος της συγκεκριμένη άσκησης είναι να υλοποιηθούν βασικές εντολές ενός απλού συνόλου εντολών (Reduced Instruction Set Computer ή RISC εντολές). Το νέο χαρακτηριστικό της συγκεκριμένης άσκησης είναι το γεγονός ότι ο υπό αναφορά καταχωρητής παρουσιάζεται σαν τελούμενο της εντολής. Το αποτέλεσμα είναι ότι χρειάζεται προσωρινή αποθήκευση της τιμής του τρέχοντος καταχωρητή και πέρασμα της τιμής στο αρχείο καταχωρητών μέσω του feedback register.

Ζητούμενες Μικροεντολές

Να γραφούν μικροπρογράμματα για την υλοποίηση των ακόλουθων μακροεντολών.

- **LOAD R, $K :** φόρτωσε στο καταχωρητή R τα περιεχόμενα της θέσης μνήμης με διευθυνση K.
- **STORE R, $K :** αποθήκευσε το καταχωρητή R στη θέσης μνήμης με διευθυνση K.
- **ADD R1, $K :** πρόσθεσε το περιεχόμενο της θέσης μνήμης με διεύθυνση K στον καταχωρητή R.
- **SUB R1, $K :** αφαίρεσε απο το περιεχόμενο του καταχωρητή R το περιεχόμενο της θέσης μνήμης με διεύθυνση K.
- **ADD R1, R2 :** πρόσθεσε τα περιεχόμενα των καταχωρητών R1 και R2.
- **SHL R :** αριστερή ολίσθηση στα περιεχόμενα του καταχωρητή R.
- **DEC R :** μείωση των περιεχόμενων του καταχωρητή R κατα ένα.
- **HALT :** τέλος εκτέλεσης του προγράμματος.

Υπόδειξη: Τα R, R1 και R2 παίρνουν τιμές από 00 (hex) μέχρι 0F (hex) και αποτελούν τελούμενα των παραπάνω εντολών.

Ζητούμενο Μακροπρόγραμμα

Να γραφεί πρόγραμμα για τον υπολογισμό της έκφρασης: W = 4 * (X + Y – Z -1) + 2 * (X + Y – 1) , όπου τα W, X, Y και Z είναι θέσεις της κύριας μνήμης. Οι τιμές των X, Y και Z θα σας δοθούν κατά την διάρκεια του εργαστηρίου.

Άσκηση 6: Υλοποίηση διανυσματικών εντολών.

Ο στόχος της συγκεκριμένη άσκησης είναι να υλοποιηθούν εντολές διανυσματικών επεξεργαστών (vector processors) χρησιμοποιώντας κλήση υπορουτινών σε επίπεδο μικροεντολών.

Ζητούμενες Μικροεντολές

Να γραφούν μικροπρογράμματα για την υλοποίηση των ακόλουθων μακροεντολών.

- **LOAD R, #K :** φόρτωσε στο καταχωρητή R τον δεκαεξαδικό αριθμό K.
- **VEM R1, R2, R3 (vector move) :** μεταφορά των περιεχομένων του διανύσματος V1 στο διάνυσμα V2. Το διάνυσμα V1 αρχίζει από την θέση μνήμης ίση με περιεχόμενο του R1, το διάνυσμα V2 αρχίζει από την θέση μνήμης ίση με περιεχόμενο του R2, ενώ και τα δύο διανύσματα έχουν μήκος ίσο με το περιεχόμενο του R3 καταχωρητή.
- **VEMINC R1, R2 (vector increase) :** αύξηση των περιεχομένων του διανύσματος V κατά ένα. Το διάνυσμα V αρχίζει από την θέση μνήμης ίση με περιεχόμενο του R1 και έχει μήκος ίσο με το περιεχόμενο του R2 καταχωρητή.

Υποδείξεις: Τα R, R1 και R2 παίρνουν τιμές από 00 (hex) μέχρι 0F (hex) και αποτελούν τελούμενα των παραπάνω εντολών. Η υλοποίηση των διανυσματικών εντολών θα πρέπει να γίνεται υποχρεωτικά με διαδοχικές κλήσεις μιας υπορουτίνας σε επίπεδο μικροπρογράμματος. Επίσης, ένα διάνυσμα V που αρχίζει από την θέση μνήμης K και έχει μήκος λ αναφέρεται σε μια περιοχή της κύριας μνήμης που αρχίζει από την θέση K και αποτελείται από συνεχόμενες (κατά αύξουσα σειρά) λ θέσεις μνήμης.

Ζητούμενο Μακροπρόγραμμα

Να γραφεί πρόγραμμα το οποίο αρχικά να μεταφέρει τα περιεχόμενα των θέσεων μνήμης από το 00 (hex) έως 0F (hex) στις θέσεις μνήμης X έως Y. Στην συνέχεια τα περιεχόμενα των θέσεων μνήμης X έως Y θα πρέπει να αυξάνονται κατά ένα. Τα X, Y θα σας δοθούν κατά την διάρκεια του εργαστηρίου.

Άσκηση 7: Υλοποίηση βασικών εντολών του Intel 8085

Ο στόχος της συγκεκριμένη άσκησης είναι να δημιουργήσουμε ένα instruction set architecture (ISA) παρόμοιο με αυτό του επεξεργαστή 8085 της εταιρίας INTEL. Ο 8085 είναι ένας 8-bits επεξεργαστής και αποτελεί ουσιαστικά την πρώτη εμπορική επιτυχία της εταιρίας INTEL. Το data-path του 8085 είναι 8-bits (ίδιο με το data path της πλακέτας του εργαστηρίου), όμως ο 8085 έχει 16-bits address space (η πλακέτα του εργαστηρίου έχει 8-bits). Για αυτό τον λόγο θα χρησιμοποιήσουμε μόνο τις 256 πρώτες θέσεις μνήμης. Αφού δημιουργήσουμε το σύνολο εντολών του 8085, στην συνέχεια θα το χρησιμοποιήσουμε για να υλοποιήσουμε μερικά βασικά προγράμματα.

Αρχικοποίηση του συστήματος

Αρχικά θα πρέπει να ορισθούν οι ακόλουθοι καταχωρητές.

- **Program counter :** μετρητής προγράμματος.
- **Αρχιτεκτονικοί καταχωρητές :** A (accumulator), B, C, D, E, H (high) και L (low). Όλοι οι καταχωρητές είναι των 8-bits. Οι καταχωρητές μπορούν να χρησιμοποιηθούν και ως ζευγάρια καταχωρητών. Τα διαθέσιμα ζευγάρια είναι τα H-L, B-C και D-E.
- **Καταχωρητής σημαιών :** ο καταχωρητής των macro-flags που υπάρχει στο σύστημα.
- **Προσωρινοί καταχωρητές :** μπορείτε να χρησιμοποιήσετε τους υπόλοιπους διαθέσιμους καταχωρητές ως προσωρινούς καταχωρητές.

Ζητούμενες Μικροεντολές

Να γραφούν μικροπρογράμματα για την υλοποίηση των ακόλουθων μακροεντολών.

- **LDA $K :** φόρτωση στον accumulator με το περιεχόμενο της θέσης μνήμης με διεύθυνση K.
- **STA $K :** αποθήκευση του περιεχομένου του accumulator στη θέσης μνήμης με διεύθυνση K.
- **MVI R, #K :** φόρτωση του καταχωρητή R με τον δεκαεξαδικό αριθμό K.
- **MOV R1, R2 :** μεταφορά του περιεχομένου του καταχωρητή R2 στον καταχωρητή R1.
- **MOV R, M :** μεταφορά του περιεχομένου της θέσης μνήμης M (βλέπε πιο κάτω) στον καταχωρητή R.

- **MOV M, R :** μεταφορά του περιεχομένου του καταχωρητή R στη θέσης μνήμης M.
- **INR R :** αύξηση κατά ένα του περιεχομένου του καταχωρητή R.
- **INX RP :** αύξηση κατά ένα του περιεχομένου του ζεύγους καταχωρητών RP.
- **DCR R :** μείωση κατά ένα του περιεχομένου του καταχωρητή R.
- **DCX RP :** μείωση κατά ένα του περιεχομένου του ζεύγους καταχωρητών RP.
- **CMP R :** σύγκριση του περιεχομένου του accumulator με το περιεχόμενο του καταχωρητή R.
- **CMP M :** σύγκριση του περιεχομένου του accumulator με το περιεχόμενο της θέσης μνήμης M.
- **CPI #K :** σύγκριση του περιεχομένου του accumulator με τον δεκαεξαδικό αριθμό K.
- **JZ $K :** κάνε branch στη διεύθυνση K (η διεύθυνση K θα είναι η νέα τιμή του program counter) αν το zero flag είναι set (Z=1).
- **JNZ $K :** κάνε branch στη διεύθυνση K (η διεύθυνση K θα είναι η νέα τιμή του program counter) αν το zero flag είναι clear (Z=0).
- **JC $K :** κάνε branch στη διεύθυνση K (η διεύθυνση K θα είναι η νέα τιμή του program counter) αν το carry flag είναι set (C=1).
- **JNC $K :** κάνε branch στη διεύθυνση K (η διεύθυνση K θα είναι η νέα τιμή του program counter) αν το carry flag είναι clear (C=0).
- **HALT :** τέλος εκτέλεσης του προγράμματος.

Υποδείξεις: Τα R, R1 και R2 είναι κάποιος από τους καταχωρητές A, B, C, D, E., H και L. Επίσης, όπου εμφανίζεται το γράμμα M σημαίνει η θέση μνήμης που δείχνει το ζευγάρι καταχωρητών H και L. Το ζευγάρι καταχωρητών H και L χρησιμοποιείται ως δείκτης στη μνήμη. Τέλος, με RP (register pair) αναφερόμαστε σε ένα από τα τρία διαθέσιμα ζευγάρια καταχωρητών του 8085.

Ζητούμενα Μακροπρογράμματα

Να γραφούν τα ακόλουθα προγράμματα.

- Εύρεση του μέγιστου μεταξύ των περιεχομένων των θέσεων μνήμης X και Y. Ο μέγιστος αριθμός θα πρέπει να τοποθετείται στην θέση μνήμης Y+1.

- Εύρεση του τετραγώνου ενός αριθμού μεταξύ 0-9 χρησιμοποιώντας lookup table. Το lookup table θα πρέπει να αποθηκευτεί στην μνήμη πριν την εκτέλεση του προγράμματος και θα πρέπει να περιέχει τα τετράγωνα τον αριθμών 0 έως 9 (σε δέκα συνεχόμενες θέσεις μνήμης). Το lookup table θα τοποθετείται στις θέσεις μνήμης X έως Y. Ο αριθμός εισόδου (που θέλουμε να υπολογίσουμε το τετράγωνοι του) θα τοποθετείται στην θέση X-1 πριν την εκτέλεση του προγράμματος, ενώ το αποτέλεσμα (τετράγωνο) θα τοποθετείται στην θέση μνήμης Y+1. Αν ο αριθμός εισόδου είναι μεγαλύτερος του 9 τότε στην θέση μνήμης Y+1, θα πρέπει να τοποθετείται το FF (hex).

Σε όλες τις περιπτώσεις τα X, Y θα σας δοθούν κατά την διάρκεια του εργαστηρίου.