ΕΡΓΑΣΤΗΡΙΟ ΣΥΓΧΡΟΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Δούρου Βασιλική Ευαγγελία- Α.Μ.:1072633- Εργαστηριακό τμήμα: Β

Πεσκελίδης Παύλος- Α.Μ.:1072483- Εργαστηριακό τμήμα: Β

**Άσκηση 3:**

Οι καταχωρητές που θα χρησιμοποιηθούν στην άσκηση, σύμφωνα με την ενημέρωση, είναι ο Accumulator:1001, ο Program Counter:1011, ο βοηθητικός καταχωρητής Χ:1110 και ο βοηθητικός καταχωρητής Υ:1101.

Οι ψευδοκώδικες για τις ζητούμενες μακροεντολές είναι οι ακόλουθοι:

Για την LDA $K,X:

PC+1→PC, MAR

MDR+0→ACC

ACC+X→NOP, MAR

MDR+0→ACC

PC+1→PC, MAR

NEXT(PC)

Αρχικά, αυξάνουμε κατά ένα τον PC για να διευθυνσιοδοτήσουμε την κύρια μνήμη για να πάρουμε το έντελο. Στη συνέχεια, αποθηκεύουμε το περιεχόμενο του MDR στον συσσωρευτή και, έπειτα, προσθέτουμε το περιεχόμενο του με το περιεχόμενο του βοηθητικού καταχωρητή Χ και διευθυνσιοδοτούμε την κύρια μνήμη για να περάσει όλο αυτό το περιεχόμενο στον MDR και μέσω αυτού στον συσσωρευτή. Τέλος, αυξάνουμε κατά ένα τον PC και διευθυνσιοδοτούμε την κύρια μνήμη με την νέα του τιμή για να αρχίσει την εκτέλεση της επόμενης εντολής και αρχίζει η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Για την LDX #K:

PC+1→PC, MAR

MDR+0→X

PC+1→PC, MAR

NEXT(PC)

Αρχικά, αυξάνουμε κατά ένα τον PC, ώστε να πάρουμε το έντελο από την κύρια μνήμη. Έπειτα, περνάμε το περιεχόμενο του MDR, που είναι ο δεκαεξαδικός αριθμός Κ, στον βοηθητικό καταχωρητή Χ. Τέλος, αυξάνουμε κατά ένα τον PC και διευθυνσιοδοτούμε την κύρια μνήμη με την νέα του τιμή για να αρχίσει την εκτέλεση της επόμενης εντολής και αρχίζει η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Για την INX:

X+1→X

PC+1→PC,MAR

NEXT(PC)

Εδώ αυξάνουμε κατά ένα το περιεχόμενο του καταχωρητή Χ και, έπειτα, αυξάνουμε κατά ένα τον PC και διευθυνσιοδοτούμε την κύρια μνήμη με την νέα του τιμή για να αρχίσει την εκτέλεση της επόμενης εντολής και αρχίζει και η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Για την CMPX #Y:

PC+1→PC,MAR

MDR+0→Y

X-Y→NOP,MSTATUSCLK

PC+1→PC,MAR

NEXT(PC)

Αρχικά, αυξάνουμε κατά ένα τον PC, ώστε να πάρουμε το έντελο από την κύρια μνήμη. Έπειτα, περνάμε το περιεχόμενο του MDR, που είναι ο δεκαεξαδικός αριθμός Υ, στον βοηθητικό καταχωρητή Υ. Στη συνέχεια, αφαιρούμε από τα περιεχόμενα του βοηθητικού καταχωρητή Χ τα περιεχόμενα του βοηθητικού καταχωρητή Υ και ανανεώνουμε τα macro-flags. Τέλος, αυξάνουμε κατά ένα τον PC και διευθυνσιοδοτούμε την κύρια μνήμη με την νέα του τιμή για να αρχίσει την εκτέλεση της επόμενης εντολής και αρχίζει και η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Για την STA $K,X:

PC+1→PC,MAR

MDR+0→Y

Y+X→NOP,MAR

ACC+0→NOP,MWE~

PC+1→PC,MAR

NEXT(PC)

Αρχικά, αυξάνουμε κατά ένα τον PC, ώστε να πάρουμε το έντελο από την κύρια μνήμη. Έπειτα, περνάμε το περιεχόμενο του MDR, που είναι ο δεκαεξαδικός αριθμός K, στον βοηθητικό καταχωρητή Υ. Έπειτα, προσθέτουμε τα περιεχόμενα του βοηθητικού καταχωρητή Χ με αυτά του βοηθητικού καταχωρητή Υ και αποθηκεύουμε το αποτέλεσμα στον MDR. Στη συνέχεια, κάνουμε enable το σήμα MWE~ για να γίνει εγγραφή στην κύρια μνήμη και, τέλος, αυξάνουμε κατά ένα τον PC και διευθυνσιοδοτούμε την κύρια μνήμη με την νέα του τιμή για να αρχίσει την εκτέλεση της επόμενης εντολής και, έπειτα, αρχίζει η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Για την ADC $K,X:

PC+1→PC,MAR

MDR+0→Y

X+Y→NOP,MAR

MDR+ACC→ACC,CARRYE~

PC+1→PC,MAR

NEXT(PC)

Αρχικά, αυξάνουμε κατά ένα τον PC, ώστε να πάρουμε το έντελο από την κύρια μνήμη. Έπειτα, περνάμε το περιεχόμενο του MDR, που είναι ο δεκαεξαδικός αριθμός K, στον βοηθητικό καταχωρητή Υ. Στη συνέχεια, προσθέτουμε τα περιεχόμενα του βοηθητικού καταχωρητή Χ και του βοηθητικού καταχωρητή Υ και τα περνάμε στον MDR, όπου ακολούθως τα προσθέτουμε με τα περιεχόμενα του accumulator, αποθηκεύοντας το αποτέλεσμα σε αυτόν, και κάνουμε enable το σήμα CARRYE~ για να γίνει η πρόσθεση με κρατούμενο. Τέλος, αυξάνουμε κατά ένα τον PC και διευθυνσιοδοτούμε την κύρια μνήμη με την νέα του τιμή για να αρχίσει την εκτέλεση της επόμενης εντολής και, έπειτα, αρχίζει η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Για την CRC:

0+0→NOP,MSTATUSCLK

PC+1→PC,MAR

NEXT(PC)

Εδώ προσθέσαμε το μηδέν με τον μηδέν, καθώς η πράξη αυτή δεν δημιουργεί κρατούμενο και ενεργοποιήσαμε το σήμα MSTATUSCLK για να ανανεωθούν τα macro-flags και να γίνει η τιμή του κρατουμένου 0. Έπειτα, αυξήσαμε κατά ένα τον PC και διευθυνσιοδοτήσαμε την κύρια μνήμη με την νέα του τιμή για να αρχίσει την εκτέλεση της επόμενης εντολής και, έπειτα, άρχισε η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Για την BRNZ $K:

PC+1→PC,MAR

MDR+0→ACC

ACC-1→PC

PC+1→PC,MAR

NEXT(PC)

Αρχικά, αυξάνουμε κατά ένα τον PC για να πάρουμε το έντελο και το αποθηκεύουμε στον accumulator. Έτσι, έχουμε τη διεύθυνση στην οποία θα γίνει το branch αποθηκευμένη. Στην περίπτωση, όμως, που το Ζ είναι ενεργοποιημένο, η επόμενη προς εκτέλεση εντολή θα είναι η PC+1→PC,MAR (θα φανεί στις 40-αδες). Έπειτα, θα περάσει η τιμή του accumulator μειωμένη κατά ένα στον PC. Η τιμή αυτή θα είναι μειωμένη κατά ένα, καθώς το επόμενο βήμα του PC+1→PC,MAR θέλουμε να πραγματοποιείται είτε είναι Ζ=0 και μπορεί να γίνει το branch είτε όχι, οπότε αν η τιμή της νέας διεύθυνσης έχει μειωθεί κατά ένα και μετά της προστεθεί και ένα θα ήταν η επιθυμητή. Έπειτα, αυξάνουμε κατά ένα τον PC και διευθυνσιοδοτούμε την κύρια μνήμη με την νέα του τιμή για να αρχίσει την εκτέλεση της επόμενης εντολής και, έπειτα, αρχίζει η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Για την SHLA:

ACC+0→ACC,SRAMU

PC+1→PC,MAR

NEXT(PC)

Αρχικά, προσθέτουμε το μηδέν στα περιεχόμενα του accumulator και γράφουμε το SRAMU για να δηλώσουμε ότι θα κάνουμε απλή αριστερή ολίσθηση (θα θέσουμε I[8:6]:111 και SH~:0 όπως θα φανεί και στις 40-αδες). Έπειτα, αυξάνουμε κατά ένα τον PC και διευθυνσιοδοτούμε την κύρια μνήμη με την νέα του τιμή για να αρχίσει την εκτέλεση της επόμενης εντολής και, έπειτα, αρχίζει η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Για την HALT:

PC+0→PC,MAR

Για αυτή την μακροεντολή απλά προσθέτουμε το μηδέν στον PC για να μείνει ίδια η τιμή του και να μην μπορεί να συνεχιστεί το πρόγραμμα.

Επίσης, ο ψευδοκώδικας του Bootstrap είναι ο ακόλουθος:

1. Switches + 0 -> PC, MAR

2. NEXT(PC)

Εδώ, αρχικοποιείται το PC ανάλογα με τις τιμές των dip switches και στη συνέχεια, αρχίζει η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Οι 40-αδες για τις παραπάνω εντολές είναι οι ακόλουθες:









Το πρόγραμμα για την εκτέλεση της ζητούμενης διαδικασίας είναι το εξής:

LDX #00

LDA $10,X

SHLA

CRC

ADC $18,X

STA $20,X

INX

CMPX #08

BRNZ $02

HALT

Αρχικά, βάζουμε το 0 στον βοηθητικό καταχωρητή Χ και, έπειτα, φορτώνουμε στον accumulator το Ζ[0]. Στη συνέχεια κάνουμε αριστερή ολίσθηση για να πολλαπλασιαστεί το περιεχόμενο του accumulator με το 2, καθαρίζουμε το κρατούμενο που προκύπτει από την ολίσθηση, και έπειτα προσθέτουμε το αποτέλεσμα με το Υ[0]. Μετά, αποθηκεύουμε το αποτέλεσμα της πρόσθεσης στο W[0] και αυξάνουμε κατά ένα το περιεχόμενο του Χ για να ελέγξουμε αν θα συνεχίσει η επανάληψη, συγκρίνοντας την τιμή του με το 8. Αν το zero flag Z=0, δηλαδή αν δεν έχει ξεπεράσει η τιμή του βοηθητικού καταχωρητή Χ το 7, καθώς θέλουμε οι επαναλήψεις να είναι από i=0 έως 7, τότε το πρόγραμμα θα επιστρέφει στην εντολή LDA $10,X (που όπως θα φανεί παρακάτω θα είναι στη θέση m02 της κύριας μνήμης). Τέλος, όταν το Χ ξεπεράσει την τιμή 7, θα σταματήσουν οι επαναλήψεις και το πρόγραμμα θα τερματιστεί.

Θα θεωρήσουμε για κάθε μακροεντολή ένα μοναδικό opcode. Έτσι, θα έχουμε LDA $K,X:00H, LDX #K:01H, INX:02H, CMPX #Y:03H, STA $K,X:04H, ADC $K,X:05H, CRC:06H, BRNZ $K:07H, SHLA:08H, HALT:09H.

Στη συνέχεια, θα θεωρήσουμε ότι το μικροπρόγραμμα για κάθε μακροεντολή ξεκινάει από την ακόλουθη διεύθυνση της μικρομνήμης:

LDA $K,X:02H

LDX #K:08H

INX:0CH

CMPX #Y:0FH

STA $K,X:14H

ADC $K,X:1AH

CRC:20H

BRNZ $K:23H

SHLA:28H

HALT:2BH

Τα περιεχόμενα του Mapper φαίνονται στον ακόλουθο πίνακα:

|  |  |  |
| --- | --- | --- |
| **Mapper** | | |
| Κώδικας εντολής | Opcode/Θέση | Περιεχόμενα |
| LDA $K,X | 00000000 | 00000010 |
| LDX #K | 00000001 | 00001000 |
| INX | 00000010 | 00001100 |
| CMPX #Y | 00000011 | 00001111 |
| STA $K,X | 00000100 | 00010100 |
| ADC $K,X | 00000101 | 00011010 |
| CRC | 00000110 | 00100000 |
| BRNZ $K | 00000111 | 00100011 |
| SHLA | 00001000 | 00101000 |
| HALT | 00001001 | 00101011 |
|  |  |  |

Τα περιεχόμενα της κύριας μνήμης είναι τα ακόλουθα:

|  |  |  |
| --- | --- | --- |
| **Main Memory** | | |
| Κώδικας εντολής | Θέση | Περιεχόμενο |
| LDX #00 |  |  |
| 00000000 | 00000001 |
| 00000001 | 00000000 |
| LDA $10,X |  |  |
| 00000010 | 00000000 |
| 00000011 | 00010000 |
| SHLA |  |  |
| 00000100 | 00001000 |
| CRC |  |  |
| 00000101 | 00000110 |
| ADC $18,X |  |  |
| 00000110 | 00000101 |
| 00000111 | 00011000 |
| STA $20,X |  |  |
| 00001000 | 00000100 |
| 00001001 | 00100000 |
| INX | 00001010 | 00000010 |
| CMPX #08 | 00001011 | 00000011 |
| 00001100 | 00001000 |
| BRNZ $02 | 00001101 | 00000111 |
| 00001110 | 00000010 |
| HALT | 00001111 | 00001001 |
| Περιοχή δεδομένων:  Z[0] | 00010000 | 00000001 |
| Z[1] | 00010001 | 00000011 |
| Z[2] | 00010010 | 00000111 |
| Z[3] | 00010011 | 00000010 |
| Z[4] | 00010100 | 00000000 |
| Z[5] | 00010101 | 00001000 |
| Z[6] | 00010110 | 00000100 |
| Z[7] | 00010111 | 00000001 |
| Y[0] | 00011000 | 00001000 |
| Y[1] | 00011001 | 00000101 |
| Y[2] | 00011010 | 00000010 |
| Y[3] | 00011011 | 00000000 |
| Y[4] | 00011100 | 00000001 |
| Y[5] | 00011101 | 00001010 |
| Y[6] | 00011110 | 00000011 |
| Y[7] | 00011111 | 00000111 |

Αποτελέσματα στην κύρια μνήμη:

m20 0a

m21 0b

m22 10

m23 04

m24 01

m25 1a

m26 0b

m27 09