ΕΡΓΑΣΤΗΡΙΟ ΣΥΓΧΡΟΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

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

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

**Άσκηση 4:**

Οι καταχωρητές που θα χρησιμοποιηθούν στην άσκηση, σύμφωνα με την ενημέρωση, είναι ο Program Counter: 0010, ο Base Register: 0100, ο Stack Pointer: 0110, ο Limit Register: 0000, ο Χ: 1000 και ο Temp: 1010. Επίσης, οι θέσεις μνήμης W, X, Y, Z που θα χρησιμοποιηθούν είναι αντίστοιχα οι ακόλουθες: 16H, 04H, 10H, 22H.

Οι ψευδοκώδικες για τις ζητούμενες μακροεντολές είναι οι ακόλουθοι:

Για την LOADBR #K:

PC+1→PC, MAR

MDR+0→BR

PC+1→PC, MAR

NEXT(PC)

Αρχικά, αυξάνουμε κατά ένα τον PC για να διευθυνσιοδοτήσουμε την κύρια μνήμη για να πάρουμε το έντελο. Έπειτα, περνάμε το περιεχόμενο του MDR, που είναι ο δεκαεξαδικός αριθμός Κ, στον καταχωρητή βάσης BR. 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 LOADSP #K:

PC+1→PC, MAR

MDR+0→SP

PC+1→PC, MAR

NEXT(PC)

Αρχικά, αυξάνουμε κατά ένα τον PC για να διευθυνσιοδοτήσουμε την κύρια μνήμη για να πάρουμε το έντελο. Έπειτα, περνάμε το περιεχόμενο του MDR, που είναι ο δεκαεξαδικός αριθμός Κ, στον καταχωρητή δείκτη SP. 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 LOADLR #K:

PC+1→PC, MAR

MDR+0→LR

PC+1→PC, MAR

NEXT(PC)

Αρχικά, αυξάνουμε κατά ένα τον PC για να διευθυνσιοδοτήσουμε την κύρια μνήμη για να πάρουμε το έντελο. Έπειτα, περνάμε το περιεχόμενο του MDR, που είναι ο δεκαεξαδικός αριθμός Κ, στον καταχωρητή ορίου LR. 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 PUSH $K:

SP-1→SP

SP-LR→NOP, MSTATUSCLK

PC+1→PC, MAR

MDR+0→NOP, MAR

MDR+0→X

SP+0→NOP, MAR

X+0→NOP, MWE~

PC+1→PC, MAR

NEXT(PC)

Αρχικά, ελαττώνουμε κατά ένα το περιεχόμενο του καταχωρητή στοίβας και συγκρίνουμε το περιεχόμενο του με το περιεχόμενο του καταχωρητή ορίου και περνάμε τις τιμές των micro-flags στα macro-flags. Έπειτα, αυξάνουμε κατά ένα το μετρητή προγράμματος για να διευθυνσιοδοτήσουμε την κύρια μνήμη για να πάρουμε το έντελο. Σε αυτή την εντολή (όπως θα φανεί και από τις 40-αδες) αν είναι ενεργοποιημένο το negative των macro-flags, δηλαδή αν δεν έχουμε άλλο χώρο στη στοίβα, η επόμενη προς εκτέλεση εντολή θα είναι η PC+1→PC, MAR. Ενώ, αν υπάρχει χώρος στη στοίβα τότε το περιεχόμενο του MDR, που είναι η διεύθυνση Κ, περνάει στον MAR και έπειτα το περιεχόμενο της διεύθυνσης αυτής περνάει στον βοηθητικό καταχωρητή Χ μέσω του MDR. Στη συνέχεια προσθέτουμε τα περιεχόμενα του SP με το 0 και αποθηκεύουμε τα αποτελέσματα στον MDR. Ακολούθως, 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 POP $K:

BR-SP→NOP, MSTATUSCLK

SP+0→NOP, MAR

MDR+0→X

PC+1→PC, MAR

MDR+0→NOP, MAR

X+0→NOP, MWE~

0+0→X

SP+0→NOP, MAR

X+0→NOP, MWE~

SP+1→SP

PC+1→PC, MAR

NEXT(PC)

Αρχικά, συγκρίνουμε τα περιεχόμενα του καταχωρητή στοίβας με τα περιεχόμενα του καταχωρητή βάσης και περνάμε τις τιμές των micro-flags στα macro-flags. Αν το αποτέλεσμα είναι αρνητικό, δηλαδή αν η στοίβα είναι άδεια, η επόμενη εντολή που θα εκτελεστεί είναι η πρώτη PC+1→PC, MAR. Αλλιώς, περνάμε το περιεχόμενο της διεύθυνσης που δείχνει ο SP στον βοηθητικό καταχωρητή Χ. Έπειτα, αυξάνουμε κατά ένα τον PC για να διευθυνσιοδοτήσουμε την κύρια μνήμη για να πάρουμε το έντελο. Αν ήταν ενεργοποιημένο από πριν το negative macro-flag τότε η επόμενη εντολή που θα εκτελεστεί θα είναι η PC+1→PC, MAR. Αλλιώς, προσθέτουμε τα περιεχόμενα του MDR με το 0 και αποθηκεύουμε το αποτέλεσμα στον MDR. Στη συνέχεια, κάνουμε enable το σήμα MWE~ για να γίνει εγγραφή στην κύρια μνήμη. Καθώς, όπως ζητείται από την εκφώνηση, θέλουμε μετά το POP να αφαιρεθεί η τιμή από τη στοίβα, αποθηκεύουμε το 0 στον βοηθητικό καταχωρητή Χ και προσθέτουμε τα περιεχόμενα του SP με το 0 και αποθηκεύουμε το αποτέλεσμα στον MDR και κάνουμε enable το σήμα MWE~ για να γίνει εγγραφή στην κύρια μνήμη. Τέλος, αυξάνουμε το SP κατά 1 για να δείξει στην επόμενη κορυφή της στοίβας 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, έπειτα, αρχίζει η διερμήνευση της επόμενης μακροεντολής, διαβάζοντας το opcode της.

Για την ADD:

SP+1→X

BR-X→NOP, MSTATUSCLK

SP+0→NOP, MAR

MDR+0→X

SP+1→NOP, MAR

MDR+X→X

SP+0→NOP, MAR

X+0→NOP, MWE~

PC+1→PC, MAR

NEXT(PC)

Αρχικά, αποθηκεύουμε στον βοηθητικό καταχωρητή Χ την τιμή SP+1 και τον συγκρίνουμε τον BR, καθώς θέλουμε να ελέγξουμε αν έχουμε τουλάχιστον δύο στοιχεία στη στοίβα, περνώντας τις τιμές των micro-flags στα macro-flags. Αν, ενεργοποιηθεί το negative macro-flag, δηλαδή αν έχουμε λιγότερα από δύο στοιχεία στη στοίβα, τότε η επόμενη προς εκτέλεση εντολή είναι η PC+1→PC, MAR (όπως θα φανεί στις 40-αδες). Αλλιώς, θα περαστεί στον καταχωρητή Χ το περιεχόμενο της διεύθυνσης που είναι αποθηκευμένη στον καταχωρητή SP και, έπειτα, θα περαστεί στον MDR το περιεχόμενο της διεύθυνσης που είναι αποθηκευμένη στον SP+1. Στη συνέχεια, θα προστεθούν αυτά τα δύο και το αποτέλεσμα θα αποθηκευτεί στον Χ και μετά θα προσθέσουμε τα περιεχόμενα του SP με το 0 και θα αποθηκεύσουμε το αποτέλεσμα στον MDR και θα κάνουμε enable το σήμα MWE~ για να γίνει εγγραφή στην κύρια μνήμη. Τέλος, θα αυξήσουμε κατά ένα τον PC και θα διευθυνσιοδοτήσουμε 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 SUB:

SP+1→X

BR-X→NOP, MSTATUSCLK

SP+0→NOP, MAR

MDR+0→X

SP+1→NOP, MAR

X-MDR→X

SP+0→NOP, MAR

X+0→NOP, MWE~

PC+1→PC, MAR

NEXT(PC)

Αρχικά, αποθηκεύουμε στον βοηθητικό καταχωρητή Χ την τιμή SP+1 και τον συγκρίνουμε τον BR, καθώς θέλουμε να ελέγξουμε αν έχουμε τουλάχιστον δύο στοιχεία στη στοίβα, περνώντας τις τιμές των micro-flags στα macro-flags. Αν, ενεργοποιηθεί το negative macro-flag, δηλαδή αν έχουμε λιγότερα από δύο στοιχεία στη στοίβα, τότε η επόμενη προς εκτέλεση εντολή είναι η PC+1→PC, MAR (όπως θα φανεί στις 40-αδες). Αλλιώς, θα περαστεί στον καταχωρητή Χ το περιεχόμενο της διεύθυνσης που είναι αποθηκευμένη στον καταχωρητή SP και, έπειτα, θα περαστεί στον MDR το περιεχόμενο της διεύθυνσης που είναι αποθηκευμένη στον SP+1. Στη συνέχεια, θα αφαιρεθούν αυτά τα δύο και το αποτέλεσμα θα αποθηκευτεί στον Χ και μετά θα προσθέσουμε τα περιεχόμενα του SP με το 0 και θα αποθηκεύσουμε το αποτέλεσμα στον MDR και θα κάνουμε enable το σήμα MWE~ για να γίνει εγγραφή στην κύρια μνήμη. Τέλος, θα αυξήσουμε κατά ένα τον PC και θα διευθυνσιοδοτήσουμε 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.

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.

Στο πρόγραμμα μας για αυτή την άσκηση, αρχικοποιούμε τις τιμές των dip switches στο 23Η καθώς από εκείνη τη θέση μνήμης θα ξεκινήσει το πρόγραμμα μας.

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









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

LOADBR #48

LOADSP #49

LOADLR #40

PUSH $22

PUSH $10

ADD

PUSH $04

SUB

POP $16

Θα θεωρήσουμε για κάθε μακροεντολή ένα μοναδικό opcode. Έτσι, θα έχουμε LOADBR #K:01H, LOADSP #K:02H, LOADLR #K:03H, PUSH:04H, POP:05H, ADD:06H, SUB:07H.

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

LOADBR #K:02H

LOADSP #K:06H

LOADLR #K:0AH

PUSH:0EH

POP:17H

ADD:23H

SUB:2DH

Τα περιεχόμενα του Mapper φαίνονται στον ακόλουθο πίνακα:

|  |  |  |
| --- | --- | --- |
| **Mapper** | | |
| Κώδικας εντολής | Opcode/Θέση | Περιεχόμενα |
| LOADBR #K | 00000001 | 00000010 |
| LOADSP #K | 00000010 | 00000110 |
| LOADLR #K | 00000011 | 00001010 |
| PUSH | 00000100 | 00001110 |
| POP | 00000101 | 00010111 |
| ADD | 00000110 | 00100011 |
| SUB | 00000111 | 00101101 |

Τα περιεχόμενα της κύριας μνήμης είναι τα ακόλουθα:

|  |  |  |
| --- | --- | --- |
| **Main Memory** | | |
| Κώδικας εντολής | Θέση | Περιεχόμενο |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Δεδομένα:  Χ | 00000100 | 00001000 |
| Υ | 00010000 | 00000010 |
| Ζ | 00100010 | 00000101 |
| Εδώ αρχίζει το πρόγραμμα:  LOADBR #48 | 00100011 | 00000001 |
| 00100100 | 01001000 |
| LOADSP #49 | 00100101 | 00000010 |
| 00100110 | 01001001 |
| LOADLR #40 | 00100111 | 00000011 |
| 00101000 | 01000000 |
| PUSH $22 | 00101001 | 00000100 |
| 00101010 | 00100010 |
| PUSH $10 | 00101011 | 00000100 |
| 00101100 | 00010000 |
| ADD | 00101101 | 00000110 |
| PUSH $04 | 00101110 | 00000100 |
| 00101111 | 00000100 |
| SUB | 00110000 | 00000111 |
| POP $16 | 00110001 | 00000101 |
| 00110010 | 00010110 |