Αναφορά 4ης εργαστηριακής άσκησης

Δημήτριος Κωστορρίζος, Α.Μ:1054419 Παναγιώτης Τσάκας, Α.Μ:1054364

Ως Base Register χρησιμοποιούμε τον καταχωρητή 3 (R3), ως Program counter τον καταχωρητή 1 (R1), ως Limit Register τον καταχωρητή 4 (R4), ως Stack Pointer τον καταχωρητή 5 (R5) και ως temp τον καταχωρητή 2(R2).

MICRO

m00 00000 000 000 111 000 011 0000 0001 00 0111010111 c//Bootstrap

m01 00000 000 000 000 000 001 0000 0000 00 0010000000 c//Bootstrap

m02 00000 000 000 101 000 011 0001 0001 01 1111011110 c//LOADBR #K,PC+1->PC,MAR

m03 00000 000 000 111 000 011 0000 0011 00 1110011101 c//MDR+0->BR

m04 00000 000 000 101 000 011 0001 0001 01 1111011110 c//PC+1->PC,MAR

m05 00000 000 000 000 000 001 0000 0000 00 0010000000 c//NEXT(PC)

m06 00000 000 000 101 000 011 0001 0001 01 1111011110 c//LOADSP #K,PC+1->PC,MAR

m07 00000 000 000 111 000 011 0000 0101 00 1110011101 c//MDR+0->SP

m08 00000 000 000 101 000 011 0001 0001 01 1111011110 c//PC+1->PC,MAR

m09 00000 000 000 000 000 001 0000 0000 00 0010000000 c//NEXT(PC)

m0a 00000 000 000 101 000 011 0001 0001 01 1111011110 c//LOADLR #K,PC+1->PC,MAR

m0b 00000 000 000 111 000 011 0000 0100 00 1110011101 c//MDR+0->LR

m0c 00000 000 000 101 000 011 0001 0001 01 1111011110 c//PC+1->PC,MAR

m0d 00000 000 000 000 000 001 0000 0000 00 0010000000 c//NEXT(PC)

m0e 00000 000 000 001 010 001 0101 0100 00 1110111111 c//PUSH $K,SP-LR->NOP,FLAGS

m0f 00110 011 010 101 000 011 0001 0001 01 1111011110 c//BRN 6, PC+1->PC,MAR

m10 00000 000 000 111 000 001 0000 0000 00 1011011101 c//(N=0) MDR+0->NOP,MAR

m11 00000 000 000 111 000 011 0000 0000 00 1110011101 c//MDR+0->temp

m12 00000 000 000 100 000 001 0101 0000 00 1011011111 c//SP+0->NOP,MAR

m13 00000 000 000 100 000 001 0000 0000 00 1000011111 c//temp+0->NOP,MWE~

m14 00000 000 000 101 001 011 0101 0101 01 1110011110 c//SP-1->SP

m15 00000 000 000 101 000 011 0001 0001 01 1111011110 c//(N=1) PC+1->PC,MAR

m16 00000 000 000 000 000 001 0000 0000 00 0010000000 c//NEXT(PC)

m17 00000 000 000 001 010 001 0011 0101 00 1110111111 c//POP $K,BR-SP->NOP,FLAGS

m18 00110 011 011 101 000 011 0001 0001 01 1110011110 c//BRZ 6, PC+1->PC

m19 00000 000 000 101 000 011 0101 0101 01 1111011110 c//(Z=0) SP+1->SP,MAR

m1a 00000 000 000 111 000 011 0000 0000 00 1110011101 c//MDR+0->temp

m1b 00000 000 000 100 000 011 0001 0001 00 1111011111 c//PC+0->PC,MAR

m1c 00000 000 000 111 000 001 0000 0000 00 1011011101 c//MDR+0->NOP,MAR

m1d 00000 000 000 100 000 001 0000 0000 00 1000011111 c//temp+0->NOP,MWE~

m1e 00000 000 000 101 000 011 0001 0001 01 1111011110 c//(Z=1) PC+1->PC,MAR

m1f 00000 000 000 000 000 001 0000 0000 00 0010000000 c//NEXT(PC)

m20 00000 000 000 101 000 011 0101 0101 01 1111011110 c//ADD,SP+1->SP,MAR

m21 00000 000 000 001 001 001 0011 0101 00 1110111111 c//SP-BR->NOP,FLAGS

m22 00010 011 010 000 000 001 0000 0000 00 1010011111 c//If BRN +2

m23 00100 010 000 000 000 001 0000 0000 00 1010011111 c//(N=0) JMP 4

m24 00000 000 000 111 000 011 0000 0000 00 1110011101 c//(N=1) MDR+0->temp

m25 00000 000 000 101 000 011 0101 0101 01 1111011110 c//SP+1->SP,MAR

m26 00000 000 000 101 000 001 0000 0000 00 1000011101 c//MDR+temp->NOP,MWE~

m27 00000 000 000 101 001 011 0101 0101 01 1110011110 c//SP-1->SP

m28 00000 000 000 101 000 011 0001 0001 01 1111011110 c//PC+1->PC,MAR

m29 00000 000 000 000 000 001 0000 0000 00 0010000000 c//NEXT(PC)

m2a 00000 000 000 101 000 011 0101 0101 01 1111011110 c//SUB,SP+1->SP,MAR

m2b 00000 000 000 001 001 001 0011 0101 00 1110111111 c//SP-BR->NOP,FLAGS

m2c 00010 011 010 000 000 001 0000 0000 00 1010011111 c//BRN 2

m2d 00100 010 000 000 000 001 0000 0000 00 1010011111 c//(N=0) JMP 4

m2e 00000 000 000 111 000 011 0000 0000 00 1110011101 c//(N=1) MDR+0->temp

m2f 00000 000 000 101 000 011 0101 0101 01 1111011110 c//SP+1->SP,MAR

m30 00000 000 000 101 001 001 0000 0000 00 1000011101 c//MDR-temp->NOP,MWE~

m31 00000 000 000 101 001 011 0101 0101 01 1110011110 c//SP-1->SP

m32 00000 000 000 101 000 011 0001 0001 01 1111011110 c//PC+1->PC,MAR

m33 00000 000 000 000 000 001 0000 0000 00 0010000000 c//NEXT(PC)

MAPPER

m00 02 c//LOADBR #K

m01 06 c//LOADSP #K

m02 0a c//LOADLR #K

m03 0e c//PUSH $K

m04 17 c//POP $K

m05 20 c//ADD

m06 2a c//SUB

m07 33 c//HALT

MAIN

m00 00 c//opcode LOADBR #K

m01 1f c//entelo

m02 01 c//opcode LOADSP #K

m03 1f c//entelo

m04 02 c//opcode LOADLR #K

m05 18 c//entelo

m06 03 c//opcode PUSH $K

m07 12 c//entelo

m08 03 c//opcode PUSH $K

m09 13 c//entelo

m0a 03 c//opcode PUSH $K

m0b 14 c//entelo

m0c 06 c//opcode SUB

m0d 05 c//opcode ADD

m0e 04 c//opcode POP $K

m0f 16 c//entelo

m10 07 c//opcode HALT

m11 00

m12 0a c//Z=10

m13 05 c//Y=5

m14 03 c//X=3

m15 00

m16 00 c//W=8

m17 00

m18 00 c//stack_limit

m19 00

m1a 00

m1b 00

m1c 00

m1d 00

m1e 00

m1f 00 c//stack_base

m20 00

Η εντολή LOADBR #K αρχικοποιεί τον base register(R3) με την τιμή K.

Η εντολή LOADLR #K αρχικοποιεί τον limit register(R4) με την τιμή K.

Η εντολή LOADSP #K αρχικοποιεί τον stack pointer με την τιμή K.

Η εντολή PUSH $K,ελέγχει αν ο σωρός είναι γεμάτος και αν δεν είναι, τοποθετεί το στοιχείο που περιέχεται στην θέση με διεύθυνση K.Στην περίπτωση όπου είναι γεμάτος, δηλαδή η διεύθυνση του limit register ισούται με αυτή του stack pointer αυξημένη κατά 1, το stack δεν επηρεάζεται και εκτελείται η επόμενη προς εκτέλεση εντολή. Στην περίπτωση, όπου υπάρχει άδεια θέση στο stack, τοποθετείται το στοιχείο που περιέχεται στην θέση με διεύθυνση K και ο stack pointer αυξάνεται κατά 1,δείχνοντας στη επόμενη άδεια θέση.

Η εντολή POP $K,ελέγχει αν ο σωρός είναι άδειος και αν δεν είναι, εξάγει το τελευταίο στοιχείο του stack,η διεύθυνση του οποίου περιέχεται στον stack pointer μειωμένο κατά 1,τοποθετώντας το στην θέση με διεύθυνση το στοιχείο που περιέχεται στην θέση με διεύθυνση K. Στην περίπτωση όπου είναι άδειος, δηλαδή η διεύθυνση του base register ισούται με αυτή του stack pointer, το stack δεν επηρεάζεται και εκτελείται η επόμενη προς εκτέλεση εντολή. Στην όπου υπάρχει τουλάχιστον ένα στοιχείο στο stack, το στοιχείο εξάγεται και τοποθετείται στη θέση με διεύθυνση το στοιχείο που περιέχεται στην θέση με διεύθυνση K και ο stack pointer μειώνεται κατά 1,δείχνοντας στη επόμενη άδεια θέση.

Η εντολή ADD, ελέγχει αν υπάρχουν τουλάχιστον 2 στοιχεία και αν υπάρχουν τα προσθέτει. Στην περίπτωση, όπου υπάρχουν λιγότερα από 2 στοιχεία, το stack δεν επηρεάζεται και εκτελείται η επόμενη προς εκτέλεση εντολή. Ειδικότερα, αν υπάρχουν 2 τουλάχιστον στοιχεία, ο stack pointer μειώνεται κατά 1,δείχνοντας στον πρώτο προσθετέο. Ο αριθμός αυτός αποθηκεύεται σε ένα καταχωρητή(temp).Έπειτα, ο stack pointer μειώνεται κατά 1,δείχνωντας στον δεύτερο προσθετέο. Ο αριθμός αυτός προστίθεται με τον πρώτο προσθετέο που περιέχεται στον καταχωρητή(temp) και αποθηκεύεται στον ίδιο καταχωρητή. Στο τέλος, το περιεχόμενο(άθροισμα) του καταχωρητή(temp) αποθηκεύεται στην θέση μνήμης της οποίας το περιεχόμενο περιέχεται στον stack pointer και ο stack pointer αυξάνεται κατά 1,δείχνοντας στη επόμενη άδεια θέση.

Η εντολή SUB, ελέγχει αν υπάρχουν τουλάχιστον 2 στοιχεία και αν υπάρχουν, αφαίρει το προτελευταίο στοιχείο από το τελευταίο. Στην περίπτωση, όπου υπάρχουν λιγότερα από 2 στοιχεία, το stack δεν επηρεάζεται και εκτελείται η επόμενη προς εκτέλεση εντολή. Ειδικότερα, αν υπάρχουν 2 τουλάχιστον στοιχεία, ο stack pointer μειώνεται κατά 1,δείχνοντας στον αφαιρέτη. Ο αριθμός αυτός αποθηκεύεται σε ένα καταχωρητή(temp).Έπειτα, ο stack pointer μειώνεται κατά 1,δείχνοντας στον αφαιρετέο. Ο αφαιρέτης, που περιέχεται στον καταχωρητή(temp), αφαιρείται από τον αφαιρετέο και αποθηκεύεται στον ίδιο καταχωρητή. Στο τέλος, ο stack pointer αυξάνεται κατά 1, το περιεχόμενο(διαφορά) του καταχωρητή(temp) αποθηκεύεται στην θέση μνήμης της οποίας το περιεχόμενο περιέχεται στον stack pointer και ο stack pointer αυξάνεται κατά 1, δείχνοντας στη επόμενη άδεια θέση.

Στο μικροπρόγραμμά μας, χρησιμοποιούμε την εντολή LOADBR #K, όπου K ο αριθμός δεκαεξαδικός 1f, για να αρχικοποιήσουμε τον base register(R3) με την τιμή 1f, την εντολή LOADLR #K, όπου K ο δεκαεξαδικός αριθμός 18, για να αρχικοποιήσουμε τον limit register(R4) με την τιμή 18 και την εντολή LOADSP #K, όπου K ο δεκαεξαδικός αριθμός 1f, για να αρχικοποιήσουμε τον base pointer(R5) με την τιμή 1f, ορίζοντας έτσι το 8-θέσεων stack(18-1f), ξεκινώντας από την διεύθυνση που περιέχεται στον base register, με τελική θέση την διεύθυνση που περιέχεται στον limit register. Έπειτα, χρησιμοποιούμε την εντολή PUSH $K,

όπου K ο αριθμός δεκαεξαδικός 0a, τοποθετώντας τον στην 1η θέση του stack και μετά χρησιμοποιούμε την εντολή PUSH $K, όπου K ο αριθμός δεκαεξαδικός 05, τοποθετώντας τον στην 2η θέση του stack. Καλούμε την εντολή SUB, αποθηκεύοντας την διαφορά(0a-05) στην 1η θέση του stack. Στην συνέχεια, χρησιμοποιούμε την εντολή PUSH $K, όπου K ο αριθμός δεκαεξαδικός 03, τοποθετώντας τον στην 2η θέση του stack. Καλούμε την εντολή ADD, αποθηκεύοντας το άθροισμα(0a-05+03) στην 1η θέση του stack. Καταλήγοντας, καλούμε την εντολή POP $K, όπου K ο αριθμός δεκαεξαδικός 16, εξάγοντας τον αριθμό(0a-05+03=08) στην θέση 16, αδειάζοντας το stack.