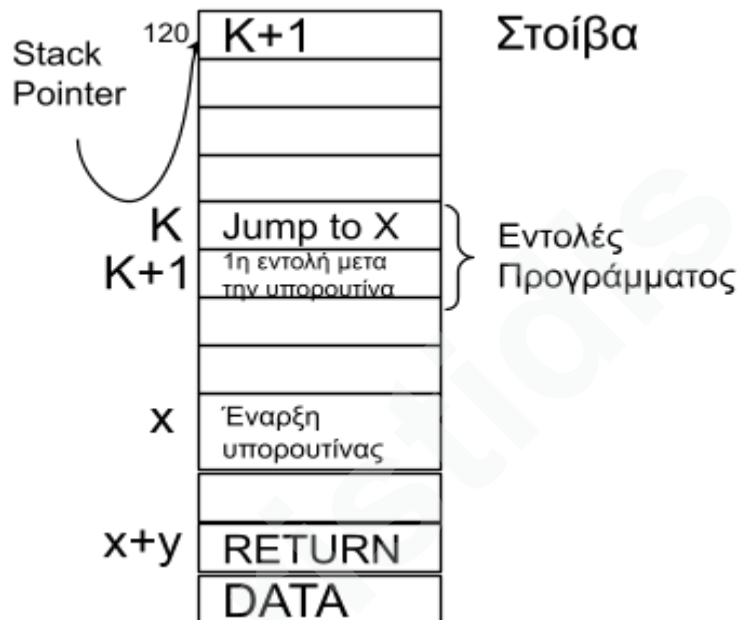


Αρχιτεκτονική Διλεξη 8

Κλήση υπορουτίνας
 Άλμα: Ο PC μετά την
 ανάκληση της Jump θα
 είναι ίσος με K+1
 (επόμενη εντολή μετά
 την υπορουτίνα) Η τιμή
 K+1 πρέπει να πάει στην
 κορυφή της στοίβας και ο
 PC να πάρει τη
 διεύθυνση της 1ης
 εντολής της υπορουτίνας



OPCODE	X
--------	---

 X= διεύθυνση της εντολής από όπου ξεκινάει η υπορουτίνα

T0: MAR ← PC, Z ← PC+1
 T1: MDR ← M[MAR], PC ← Z
 T2: IR ← MDR
 T3: Z ← SP-1
 T4: MAR ← Z, SP ← Z
 T5: MDR ← PC
 T6: M[MAR] ← MDR
 T7: PC ← IR(Address)
 RETURN
 T3: MAR ← SP
 T4: Z ← SP+1
 T5: MDR ← M[MAR], SP ← Z
 T6: PC ← MDR

MAR ← K (Θέση εντολής Jump), Z ← K+1

MDR ←

OPCODE	X
--------	---

, PC ← K+1

IR ←

OPCODE	X
--------	---

Z ← 119

MAR ← 119, SP ← 119

MDR ← K+1

M[119] ← K+1

PC ← X επειδή το τ2 αποθηκεύεται το x στον IR διαφορετικά θα είχε χαθεί το X στο βήμα T5

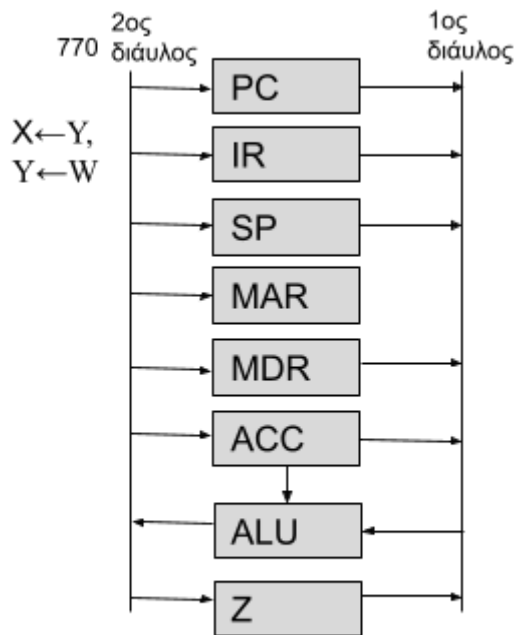
MAR ← 119

Z ← 120

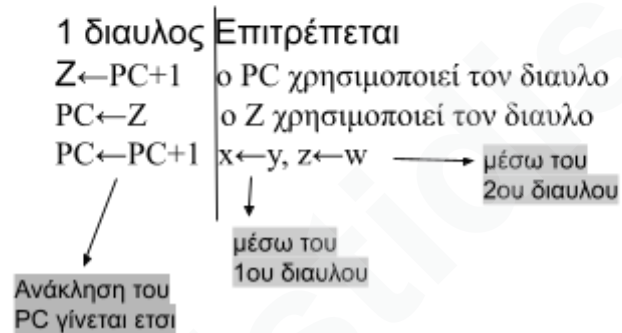
MDR ← M[119] = K+1

PC ← K+1

→ Ο PC ← X, αρα η CPU θα ζητήσει την ανάκληση της εντολής στη διεύθυνση X η οποία είναι η 1η εντολή της υπορουτίνας



Στην ανάκληση: Ο PC βάζει την τιμή του στον 1ο διάυλο, η τιμή μέσου διαύλου πάει στον αθροιστή (μέσα στην ALU) το αποτέλεσμα δεν περνάει από τον Z αλλά μέσα στον ίδιο χρόνο μεταφέρεται μέσω του 2ου διαύλου πίσω στον PC.



ADD A,B: $A \leftarrow A + B$

Με έναν διάυλο	Με 2 διαύλους
T0: $MAR \leftarrow PC, Z \leftarrow PC + 1$ T1: $MDR \leftarrow M[MAR], PC \leftarrow Z$ T2: $IR \leftarrow MDR$ T3: $MAR \leftarrow IR[Address\ 1]$ T4: $MDR \leftarrow M[MAR]$ T5: $ACC \leftarrow MDR$ T6: $MAR \leftarrow IR[Address\ 2]$ T7: $MDR \leftarrow M[MAR]$ T8: $Z \leftarrow MDR + ACC$ T9: $MDR \leftarrow Z$ T10: $MAR \leftarrow IR[Address\ 1]$ T11: $M[MAR] \leftarrow MDR$	T0: $MAR \leftarrow PC$ T1: $MDR \leftarrow M[MAR], PC \leftarrow PC + 1$ T2: $IR \leftarrow MDR$ T3: $MAR \leftarrow IR[Address\ 1]$ T4: $MDR \leftarrow M[MAR]$ T5: $ACC \leftarrow MDR, MAR \leftarrow IR[Address\ 2]$ T6: $MDR \leftarrow M[MAR]$ T7: $MDR \leftarrow MDR + ACC$ T8: $MAR \leftarrow IR[Address\ 1]$ T9: $M[MAR] \leftarrow MDR$

Ανάκληση
T0: MAR ← PC, Z ← PC+1
T1: MDR ← M[MAR], PC ← Z
T2: IR ← MDR[OPCODE], F ← 1

(το F γίνεται 1
για να αρχίσει η
εκτέλεση)

Εκτέλεση STA
T3: MAR ← MDR[ADDRESS 1]
T4: MDR ← ACC
T5: M[MAR] ← MDR

Εντολή LDA
T3: MAR ← MDR[ADDRESS 1]
T4: MDR ← M[MAR]
T5: ACC ← MDR, IR (Interact Enable=0)
then F ← 0 else G ← 1

Εκτέλεση ADD
T3: MAR ← MDR[ADDRESS 1]
T4: MDR ← M[MAR]
T5: Z ← ACC + MDR
T6: ACC ← Z

Κύκλος διακοπής
T0: Z ← SP-1
T1: SP ← Z, MAR ← Z
T2: MDR ← PC
T3: M[MAR] ← MDR
T4: MAR ← Z
T5: MDR ← M[MAR]
T6: PC ← MDR, F ← 0, G ← 0, Ien ← 0

Εκτέλεση ISR
T3: Z ← SP-1
T4: SP ← Z, MAR ← Z (Address)
T5: Z ← MDR
T6: MDR ← PC
T7: M[MAR] ← MDR, PC ← Z

F=0 κατά την ανάκληση

F=1 κατά την εκτέλεση

G=0 σε κανονική λειτουργία

G=1 αν ακολουθεί κύκλος διακοπή (interrupt)

Interact Enable=1 αν η CPU δεχτεί σήμα διακόπτη

IF(Ienable=0) then F=0 else G=1

Αρα IEN=0 δεν έχει έρθει σήμα διακόπτη αρα το F γίνεται 0 για να ξεκινήσει η CPU την ανάκληση της επόμενης εντολής (θεωρούμε για απλουστευση ότι η διακοπή μπορεί να ζητηθεί όταν ολοκληρωθεί 1 εντολή)

Αν Ien=1 τότε G=1(διακοπή)

T0,T1,T2,T3: ο PC αποθηκεύεται στην κορυφή της στοίβας για να γνωρίσει η CPU από ποιά εντολή θα συνεχίσει μόλις ολοκληρωθεί η διακοπή

T4,T5,T6:Επιστροφή της τιμής από την κορυφή της στοίβας στον PC
μετά θα βάλω το κύκλο διακοπής.

Μοναδα ελεγχου: στέλνει σήματα ελέγχου στους καταχωρητές για να γνωρίζω κάθε χρονικό παλμο Ti αν πρέπει να γράψουν η να διαβάσουν απο τον διαυλο

Τρόποι ελέγχου:

1. Προκατασκευασμένος ελέγχος: Με πύλες
2. Μικροπρογραμματιζόμενος: Μέ μνήμη