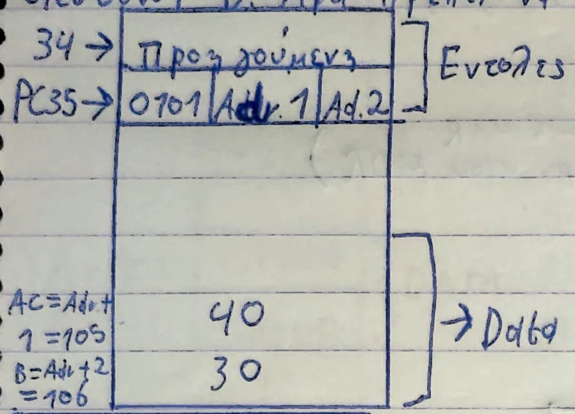
Αρχιτεκτονική Διάλεξη 7

A=A+B Ο MDR θα χρειαστεί να φέρει δεδομένα απο την μνήμη Αρα όταν θα φερει την τιμή 4 ο απο την A θα χασει την διευθυνσή B. Αρα πρεπει να αποθηκεύτει ~~σε~~ έναν βοηθητικό καταχωρητή R



• Όταν ανακλήθηκε η εντολή της δ. 34 ο PC ← 35

OPC	Ad. 1	Ad 2

Ανάκληση

T0: MAR ← PC, Z ← PC+1	MAR ← 35, z = 36
T1: MDR ← M[MAR], PC ← Z	MDR ← |0101|A|B|
T2: IR ← MDR[OPCODE]	IR ← 0101
T3: R ← MDR, MAR ← MDR[Ad 1]	R ← |0101|A|B|
T4: MDR ← M[MAR]	MDR ← 40
T5: ACC ← MDR	ACC ← 40
T6: MAR ← R(Ad2)	MAR = 106
T7: MDR ← M[MAR]	MDR ← 30 Στο σημειο αυτο κάνει την δ.Β την οποία θα παρουμε αυτο το R
T8: Z ← ACC + MDR	
T9: MDR ← Z	MDR ← 70

• ο MAR διαβάζει απο τον MDR μια διευθ. Ο MAR εχει μέγεθος όσο το μήκος της δ. αρα διαβαζει μια δ. Την δ.A μπορει να την διαβασει απο τον MDR αλλα την 2η [illegible] φορα δεν μπορεί

Γενικά:

1) Εντολή 2 παραγόντων = Αν έχω 2 μεταφορές - διαβασματα απο την μνήμη

R ← MDR: κατα την ανακλησή, ο MDR περιεχει τις δ (2) επειδή θα γίνουν 2 αναγνώσεις, οι δ. πρέπει να σωθούν R ← MDR αμέσως μετα την ανάκληση

2) Φέρνω την λέξη A (2 χρόνοι, ενημέρωση MAR, εγγραφή δεδομένων στον MDR)
ACC←MDR   2 αναγνώσεις δεδομένων. Αν ο MDR ~~περιέχει [illegible]~~ μέτα την
1η δ. δεν αποθηκευτεί, η CPU θα χάσει δεδομένα

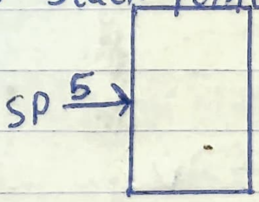
3) Φέρνω την λέξη B (2 χρόνος, ενημέρωσή MAR, εγγραφή δεδομένων στον MDR)
4) προσθέτω
5) Δίνω το αποτέλεσμα στο MDR
6) Σε 2 χρόνους γράφω στην μνήμη $\left(\begin{array}{l}\text{ενημέρωσή MAR}\\ \text{εγγραφή στον MDR}\end{array}\right)$

• C = A+B

OPC	Ad 1	Ad 2	Ad 3

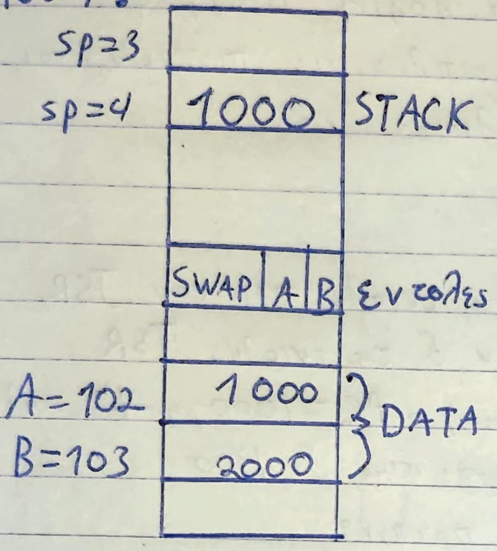
T0 MAR←PC, Z←PC+1	
T1 MDR←M[MAR], PC←Z	MDR ← $\boxed{OP}\boxed{A}\boxed{B}\boxed{C}$
T2: IR←MDR ⊛	Το IR διαβάζει απο τον MDR, άρα OPCODE και τις A, B, C δεν χρειάζεται R←MDR
T3: MAR←IR[Ad1]	Εναλλακτικά απο τον MDR
T4: MDR←M[MAR]	Ο MDR περιέχει την τιμή της δ. A.
T5: ACC←MDR	
T6: MAR←IR[Ad2]	Ο MDR περιέχει την τιμή της δ. B
T7: MDR←M[MAR]	
T8: Z←MDR+ACC	⊛ Όταν δεν μπορώ να χρησιμοποιήσω βοηθητικούς
T9: MDR←Z	καταχωρητές, χρησιμοποιώ τον IR άρα έχει μήκος όσο η δ.
T10: MAR←IR[Ad3]	
T11: M[MAR]←MDR	

• Στοίβα - Stack: Μνήμη που χρησιμοποιείται για αποθήκευση ειδικών τιμών.
• CPU → Επικοινωνεί με την στοίβα μέσω του stack Pointer → SP
• SP: Δείχνει στην τελευταία γεματή θέση.
Αν. SP←5 κενές είναι οι θέσεις 4-0.
Ισχύει: Για γράψιμο: SP-1, Για ανάγνωση SP+2



Π.χ. Να υλοποιήσετε την εντολή SWAP(A,B) η οποία εναλλάσει τις τιμές που αποθηκεύονται στις δ. A, B

Λύση:



1) Θα ζητήσω την διεύθυνση A
2) Θα φέρω τα δεδομένα στο MDR
3) Θα ζητήσω την κορυφή της στοίβας
4) Θα γράψω το A στη στοίβα
5) Θα ζητήσω το B
6) Θα στείλω το B → A
7) Θα πάρω την τιμή απο την στοίβα
8) Θα βγάλω την τιμή της στοίβας στοίβα → B

• Για να ζητήσω θέση μνήμης δίνω στον MAR την τιμή της δ. Τυπικά απο τον MDR ή IR η απο καποιον καταχωρητή R

Για να ζητήσω στοίβα: MAR ← SP

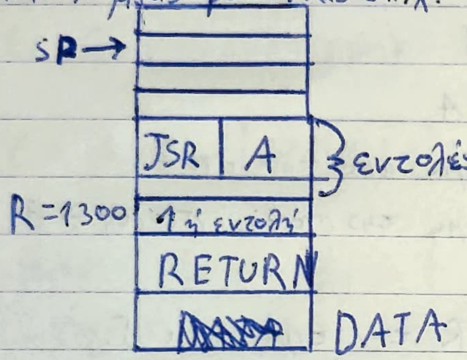
Βήματα:

T0: MAR ← PC, Z ← PC+1	
T1: MDR ← M[MAR], PC ← Z	MDR ← $\boxed{\text{SWAP} \,\vert\, A \,\vert\, B}$
T2: IR ← MDR	IR ← $\boxed{\text{SWAP} \,\vert\, A \,\vert\, B}$
T3: MAR ← IR[Ad1]	
T4: MDR ← M[MAR], Z ← SP-1	ο εσωτερικός διαυλός είναι ελευθερος, αρα ξεκινάω την μειωση.
T5: SP ← Z, MAR ← Z	Ο SP → 3 (τελευταια γεματη θεση), ο MAR ζηταει την 3 (η θεση μνημης 3 στην στοιβα εχει το 1000)
T6: M[MAR] ← MDR	
T7: MAR ← IR[Ad2]	MAR ← B
T8: MDR ← M[MAR]	MDR = 2000
T9: MAR ← IR[Ad1]	
T10: M[MAR] ← MDR	Η θ. μνημης
T11: MAR ← SP, Z ← SP+1	* απο πριν
T12: MDR ← M[MAR], SP ← Z	SP: 4 MDR: 1000
T13: MAR ← IR(Ad2)	
T14: M[MAR] ← MDR	B = 1000

Ο Stack Pointer εχει μήκος όσο και η δ. μνημης

* ● T11: Ο MAR διαβάζει την τιμή SP=3 για να πάμε να διαβάσουμε από την κορυφή της στοίβας. Όμως όταν γίνεται ανάγνωση SP←SP+1 για να δείχνει στην επόμενη θέση. Πρόκειται για λογική διαγραφή. Η τιμή 1000 της θέσης 3 δεν διαγράφεται, απλώς δεν προσπελάζεται. Όλη η στοίβα διαγράφεται στο τέλος της εκτέλεσης

Κλήση μιας ρουτίνας (π.χ. X=sum(A,B))



Όταν ανακληθεί η εντολή πριν τον JSR, ο PC δείχνει στην δ. της εντολής JSR (Jump to SubRoutine) ο PC=1000, π.χ. 1η εντολή της υπορουτίνας είναι στην δ. 1300

Για να γίνει άλμα πρέπει:

1) Η τιμή του PC (μετά την ανάλυση της) JSR πρέπει να αποθηκευτεί γιατί μετά την ρουτίνα θα ~~[illegible]~~ γυρίσουμε εκεί δηλ στην εντολή της δ. 1001

2) Το πρόγραμμα πρέπει να μεταβεί στην δ. A, αλλά για να γίνει αυτό, πρέπει PC = A

T0: MAR←PC, Z←PC+1	MAR=1000, Z=1001
T1: MDR←M[MAR], PC←Z.	MDR = | JSR | 1300 |
T2: IR←MDR	IR = | JSR | 1300 |
T3: Z←SP-1	Z=SP-1=4-1=3
T4: MAR←Z, SP←Z	
T5 MDR←PC	Ο MDR διαβάζει την τιμή PC=1001 για να την δώσει στην στοίβα
T6 M[MAR]←MDR, PC←IR(Adress)	M[3] = 1001 (η τιμή 1001 πρ στην στοίβα) | Ο PC λαμβάνει την τιμή 1300 για να ξεκινήσει το άλμα

1) πρέπει να γράψουμε τον PC στη στοίβα. Άρα να μειώσουμε το SP και να περάσουμε την μειωμένη τιμή στον MAR για να αποκωδικοποιηθεί η δ. της στοίβας

2) Ο PC να πάρει στην στοίβα μέσω του MDR

3) Γράψιμο από MDR στην στοίβα (Εξωτερικός δίαυλος) και ο PC←A για να πάει το πρόγραμμα στην υπορουτίνα