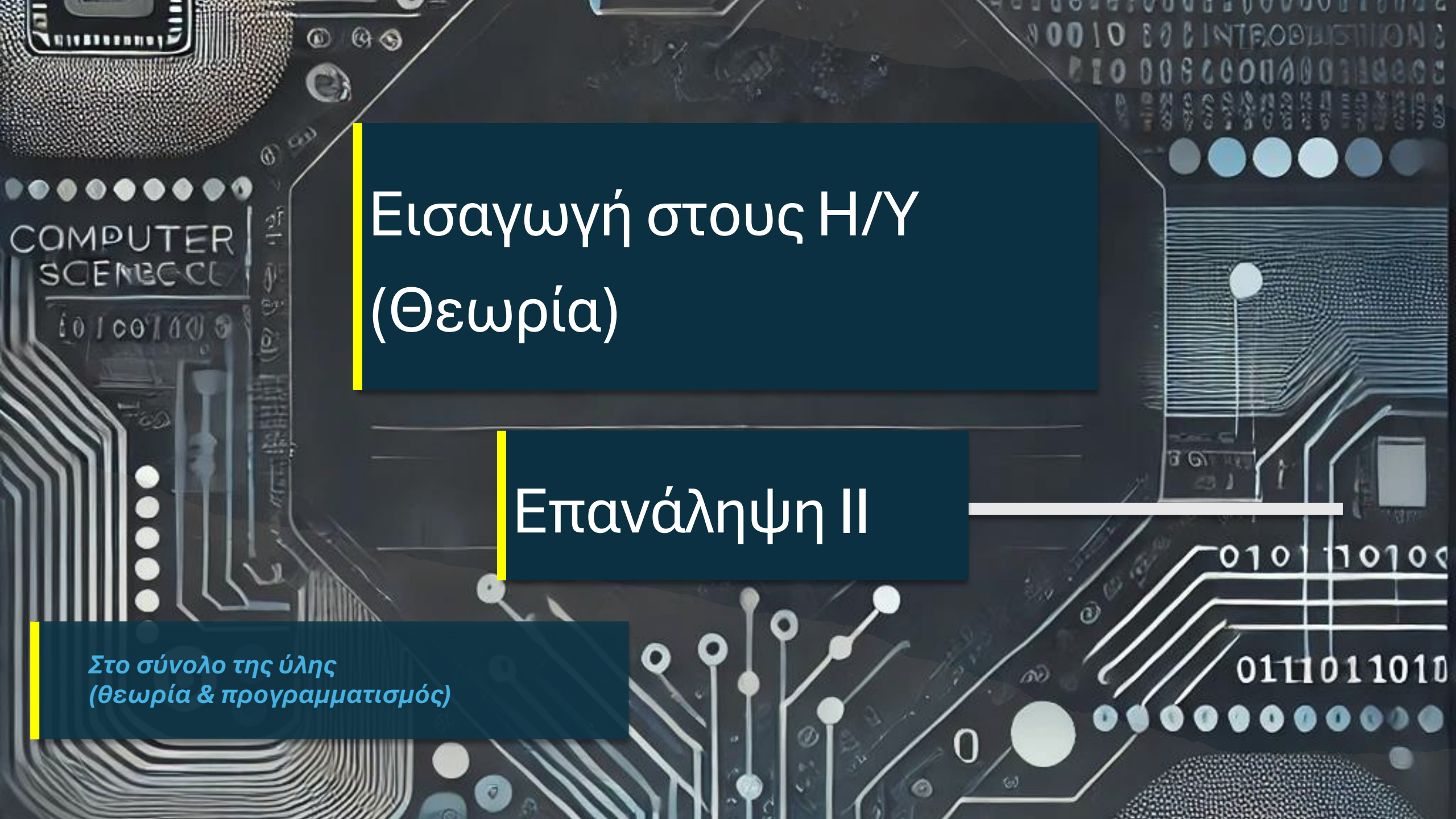


Εισαγωγή στους Η/Υ (Θεωρία)

Επανάληψη ΙΙ

Στο σύνολο της ύλης
(θεωρία & προγραμματισμός)

Γλώσσα μηχανής

Παρακάτω παρουσιάζεται ο απλός υπολογιστής που χρησιμοποιήθηκε στο μάθημα καθώς και όλο το ρεπερτόριο εντολών του. Θέλουμε ο απλός υπολογιστής μας να αφαιρεί 2 **ακεραίους** Α και Β εκφρασμένους σε συμπλήρωμα ως προς δύο (**c2**) και να παράγει και αποθηκεύει το αποτέλεσμα C. Θεωρούμε ότι ο Α αποθηκεύεται στη θέση μνήμης $64_{(10)}$ και ο Β στην $65_{(10)}$, ενώ το αποτέλεσμα C θα αποθηκευτεί στη θέση μνήμης $FD_{(16)}$. Να γραφεί το πρόγραμμα που θα κάνει την πράξη:

$$C \leftarrow A - B$$

Το πρόγραμμα να γραφεί συμπληρώνοντας τον πίνακα περιγράφοντας και την κάθε εντολή ξεχωριστά.

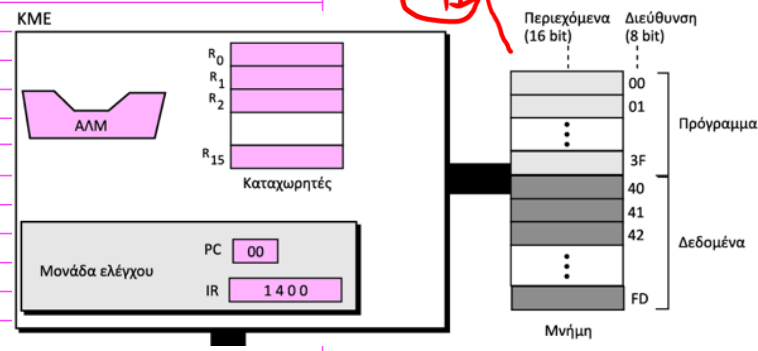
Γλώσσα Μηχανής Λύση

(#)	ΕΝΤΟΛΗ	Assembly	Επεξήγηση
1	174Ø	LOAD R7,40	φέρνω το Α στον R_7
2	1141	LOAD R1,41	φέρνω το Β στον R_1
3	611Ø	NOT R_1,R_1	Υπολογίζω C1 του Β
4	A1ØØ	INC R_1	—//— C2 του Β (Δηλαδή -Β)
5	3317	ADDI R_3,R_1,R_7	Αθροίζω A+(-B)
6	2FD3	STORE FD,R_3	Αποθήκευση στη μετ. C
7	0000	HALT	τέλος προγράμματος
8			
9			

C = A - B

A: $64_{(10)}$
B: $65_{(10)}$
C: $FD_{(16)}$

Εντολή	Κωδ. d_1	Τελεστέοι d_2	d_3	d_4	Ενέργεια
HALT	0				Διακόπτει την εκτέλεση του προγράμματος
LOAD	1	R_D	M_S		$R_D \leftarrow M_S$
STORE	2	M_D		R_S	$M_D \leftarrow R_S$
ADDI	3	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1} + R_{S2}$
ADDF	4	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1} + R_{S2}$
MOVE	5	R_D	R_S		$R_D \leftarrow R_S$
NOT	6	R_D	R_S	Ø	$R_D \leftarrow R_S$
AND	7	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1}$ AND R_{S2}
OR	8	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1}$ OR R_{S2}
XOR	9	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1}$ XOR R_{S2}
INC	A	R			R ← R + 1
DEC	B	R			R ← R – 1
ROTATE	C	R	n	0 ή 1	Rot_n R
JUMP	D	R	n		Αν $R_0 \neq$ R τότε PC = *n*, διαφορετικά συνέχισε
Υπόμνημα	R_S, R_{S1}, R_{S2}: Δεκαεξαδική διεύθυνση της θέσης των καταχωρητών **προέλευσης**				
	$R_{D\ (DESTINATION)}$: Δεκαεξαδική διεύθυνση της θέσης του καταχωρητή **προορισμού**				
	$M_{S\ (SOURCE)}$: Δεκαεξαδική διεύθυνση της θέσης μνήμης **προέλευσης**				
	M_D: Δεκαεξαδική διεύθυνση της θέσης μνήμης **προορισμού**				
	n: δεκαεξαδικός αριθμός				
	d_1, d_2, d_3, d_4: 1ο, 2ο, 3ο, και 4ο δεκαεξαδικό ψηφίο				



C = A – B = A + (-B)

↓

$C_1 + 1$

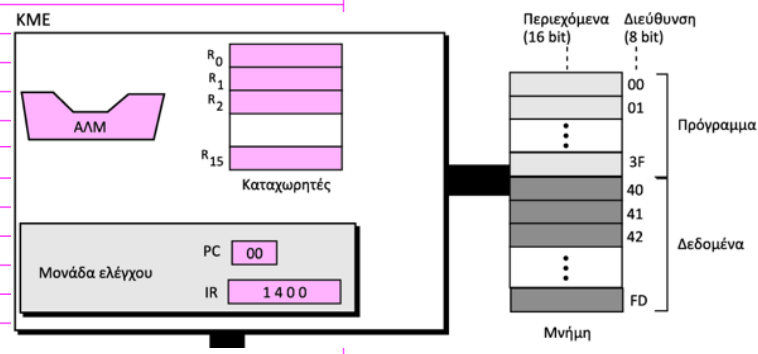
Γλώσσα Μηχανής

(#)	ΕΝΤΟΛΗ	Assembly	Επεξήγηση
1	1040	LOAD R_0, 40	είχα βέρη την $40_{(16)}$
2	1141	LOAD R_1, 41	
3	6110	NOT R_1, R_1	
4	A100	INC R_1	
5	3201	ADDI R_2, R_0, R_1	
6	1342	LOAD R_3, 42	φέρνω στην $42_{(16)}$
7	3430	ADDI R_4, R_3, R_0	
8	2F04	STORE F0, R_4	
9	0000	HALT	

Γράψτε την αντίστοιχη Assembly για το υπάρχον πρόγραμμα σε γλώσσα μηχανής και συμπληρώστε το ώστε να προσθέτει τα περιεχόμενα της θέσης μνήμης $64_{(10)}$, με αυτά της $42_{(16)}$ και να αποθηκεύει το αποτέλεσμα στη θέση μνήμης $F0_{(16)}$.

$40_{(10)}$

Εντολή	Κωδ.	Τελεστέοι			Ενέργεια
	d_1	d_2	d_3	d_4	
HALT	0				Διακόπτει την εκτέλεση του προγράμματος
LOAD	1	R_D	M_S		$R_D \leftarrow M_S$
STORE	2	M_D		R_S	$M_D \leftarrow R_S$
ADDI	3	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1} + R_{S2}$
ADDF	4	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1} + R_{S2}$
MOVE	5	R_D	R_S		$R_D \leftarrow R_S$
NOT	6	R_D	R_S		$R_D \leftarrow R_S$
AND	7	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1}$ AND R_{S2}
OR	8	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1}$ OR R_{S2}
XOR	9	R_D	R_{S1}	R_{S2}	$R_D \leftarrow R_{S1}$ XOR R_{S2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ή 1	Rot_n R
JUMP	D	R	n		Αν $R_0 \neq R$ τότε PC = *n*, διαφορετικά συνέχισε
Υπόμνημα	R_S, R_{S1}, R_{S2}: Δεκαεξαδική διεύθυνση της θέσης των καταχωρητών **προέλευσης**				
	$R_{D\ (DESTINATION)}$: Δεκαεξαδική διεύθυνση της θέσης του καταχωρητή **προορισμού**				
	$M_{S\ (SOURCE)}$: Δεκαεξαδική διεύθυνση της θέσης μνήμης **προέλευσης**				
	M_D: Δεκαεξαδική διεύθυνση της θέσης μνήμης **προορισμού**				
	n: δεκαεξαδικός αριθμός				
	d_1, d_2, d_3, d_4: 1ο, 2ο, 3ο, και 4ο δεκαεξαδικό ψηφίο				



Γλώσσα Μηχανής & C

Έχουμε τρεις κώδικες C και τους αντίστοιχους κώδικες γλώσσας μηχανής που έχουν προκύψει από το compilation των κωδίκων της C. Ταιριάξτε τους κώδικες βάσει τον αποσπασμάτων που παρουσιάζονται:

```
register short int a;
…
a = c;
…
a++;
…
b = a;
```

A

```
do {
   …
   q = calc(x,y);
   …
   ++q;
   …
} while (q != z);
```

B

```
int a = 100, b = 200;
…
int c = a + b;
…
if (c == d) {
   …
}
```

Γ

```
LOAD R4, 0xE7
…
INC R4
…
JUMP R4, 0x08
```

1

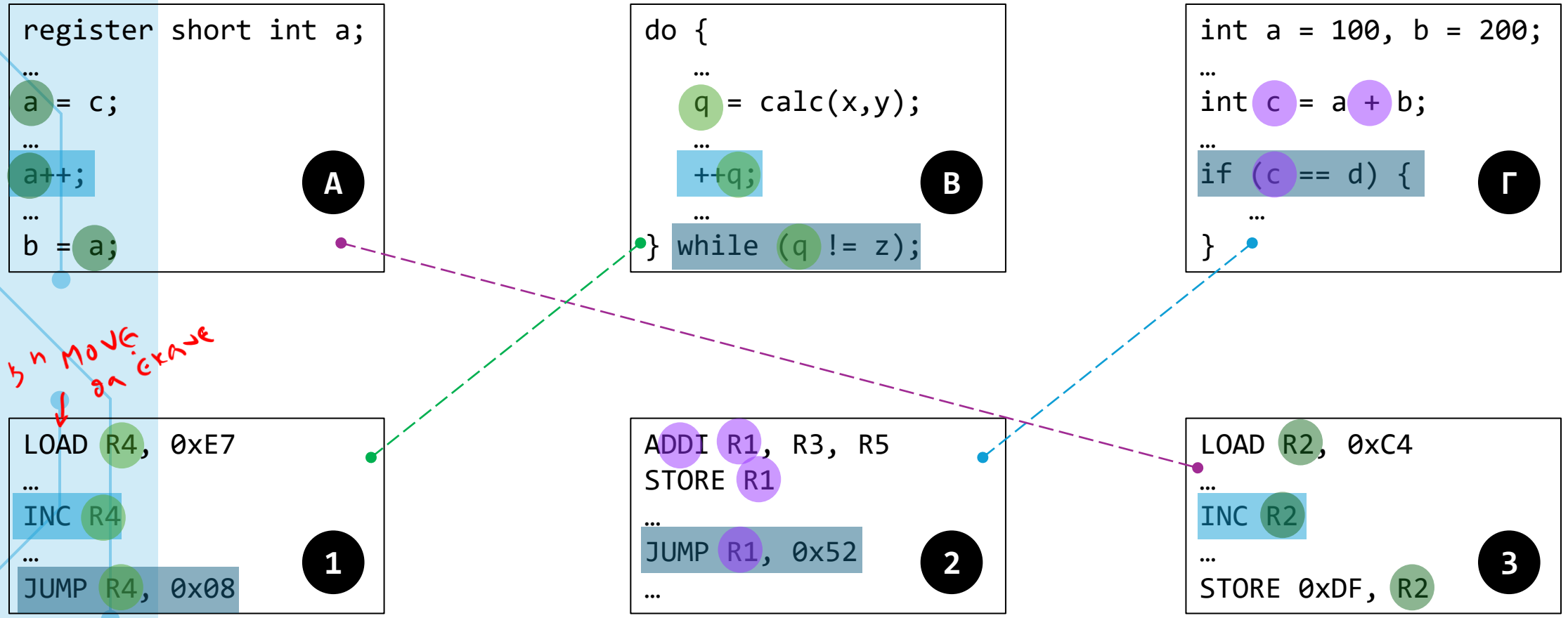
```
ADDI R1, R3, R5
STORE R1
…
JUMP R1, 0x52
…
```

2

```
LOAD R2, 0xC4
…
INC R2
…
STORE 0xDF, R2
```

3

Γλώσσα Μηχανής & C



Συγγραφή κώδικα C από το μηδέν

Γράψτε μία από τις συναρτήσεις:

- **`sum(N,arr)`** που αθροίζει τα στοιχεία ενός μονοδιάστατου πίνακα
- **`max(N, arr)`** που επιστρέφει τη μέγιστη τιμή των στοιχείων ενός μονοδιάστατου πίνακα
- **`cumsum(N, arr)`** που επιστρέφει πίνακα με το σωρευτικό άθροισμα των στοιχείων του πίνακα
- **`secSinceMidnight(h,m,s)`**
- **`seconds2HMS(sec)`** → HHMMSS (6ψήφιος ακέραιος)
- **`bytes2KiB(bytes)`**
- **`bytes2MiB(bytes)`**
- **`bytes2kB(bytes)`**
- **`bytes2MB(bytes)`**