Βασιλείου Δημήτριος A.M.: el19830
Ζαρίφης Στυλιανός A.M.: el20435
19/04/2022

Συστήματα Μικροϋπολογιστών

6ο Εξάμηνο 2021 – 2022

Ομάδα Ασκήσεων 1

Δημήτριος Βασιλείου – el19830

Στυλιανός Ζαρίφης – el20435

Table of Contents

Βασιλείου Δημήτριος A.M.: el19830
Ζαρίφης Στυλιανός A.M.: el20435
19/04/2022

Άσκηση 1

Αποκωδικοποίηση (Disassembly)

Με βάση τον πίνακα 2 του παραρτήματος των σημειώσεων έχουμε την ακόλουθη αποκωδικοποίηση του προγράμματος από γλώσσα μηχανής σε γλώσσα Assembly. Στα αριστερά φαίνεται η αποκωδικοποίηση του προγράμματος, γραμμή προς γραμμή, με τις πραγματικές διευθύνσεις στις εντολές διακλάδωσης. Στα δεξιά φαίνεται η Assembly που θα γράφαμε για να λειτουργήσει το πρόγραμμα στο μικροεπεξεργαστή 8085, δηλαδή με συμβολικές διευθύνσεις – ετικέτες σε κάθε εντολή διακλάδωσης:

```
0800  06    MVI B,01H     MVI B,01H
0801  01
0802  3A    LDA 2000H     LDA 2000H
0803  00
0804  20
0805  FE    CPI 00H       CPI 00H
0806  00
0807  CA    JZ 0813       JZ LABEL1
0808  13
0809  08
                          LABEL3:
080A  1F    RAR           RAR
080B  DA    JC 0812       JC LABEL2
080C  12
080D  08
080E  04    INR B         INR B
080F  C2    JNZ 080A      JNZ LABEL3
0810  0A
0811  08
                          LABEL2:
0812  78    MOV A,B       MOV A,B

                          LABEL1:
0813  2F    CMA           CMA
0814  32    STA 3000H     STA 3000H
0815  00
0816  30
0817  CF    RST 1         RST 1

                          END
```

Βασιλείου Δημήτριος A.M.: el19830
Ζαρίφης Στυλιανός A.M.: el20435
19/04/2022

Λειτουργία του προγράμματος

Η λειτουργία του προγράμματος είναι η εξής:

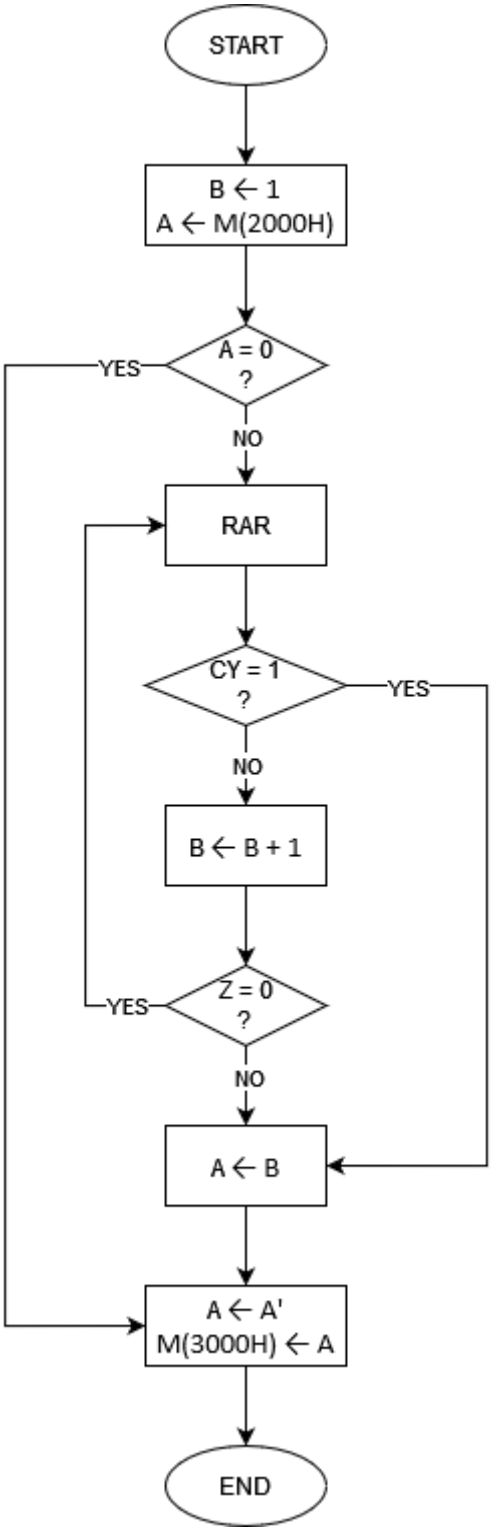
Ξεκινάμε με τον καταχωρητή B να έχει την τιμή 1 και τον A να λαμβάνει το input του χρήστη.

1. Αν ο A έχει τιμή μηδέν, μεταπηδάμε στο LABEL1, οπότε και τυπώνουμε το συμπλήρωμα της τιμής του A (Οπότε και εμφανίζεται αναμμένο LED για κάθε άσσο και σβηστό για κάθε μηδενικό της εξόδου, δηλαδή όλα τα LEDs σβηστά αφού ο A έχει τιμή 00H)
2. Αλλιώς, συνεχίζουμε στο LABEL3 όπου περιστρέφουμε τα bits του A δεξιά (συμμετέχει και το flag CY), έως ότου ένας άσσος μπει στη θέση του CY.
    a. Όσο το περιεχόμενο του CY παραμένει 0, προσαυξάνουμε το περιεχόμενο του B και συνεχίζουμε.
    b. Αν βρεθεί άσσος στο CY, μεταπηδάμε στο LABEL2
3. Από το LABEL2, εκχωρούμε την τιμή του B στον A, υπολογίζουμε το συμπλήρωμά του και το τυπώνουμε.

Εξήγηση: Παρατηρούμε πως η τιμή του B αυξάνεται κάθε φορά που περιστρέφουμε το input έως ότου βρεθεί άσσος. Άρα η έξοδος είναι η θέση του Least Significant Bit που είναι άσσος (με το λιγότερο σημαντικό bit να βρίσκεται στην πρώτη θέση).

Διάγραμμα Ροής

Στη συνέχεια, παραθέτουμε το διάγραμμα ροής του προγράμματος:



Ατέρμονη Επανάληψη

Για συνεχή λειτουργία του προγράμματος αρκεί να προσθέσουμε μια εντολή διακλάδωσης (χωρίς συνθήκη), ακριβώς πριν το τέλος του προγράμματος, ώστε να αναγκάζει τον PC να μεταπηδά στην αρχή του προγράμματος, δηλαδή στη θέση μνήμης 0800H (σε Assembly προσθέτουμε την ετικέτα START πριν την πρώτη εντολή του κώδικα). Μετά την αλλαγή, έχουμε το ακόλουθο τελικό πρόγραμμα (αρχείο `assignment1_ex1.8085`):

```
START:
      MVI B,01H   ; B = 1
      LDA 2000H   ; A = M(2000H)
      CPI 00H     ; If (A == 0) then Z = 1 else Z = 0
      JZ LABEL1   ; If (Z == 1) then goto LABEL1

LABEL3:
      RAR         ; Rotate A Right (with CY)
      JC LABEL2   ; If (CY == 1) then goto LABEL2
      INR B       ; B = B + 1
      JNZ LABEL3  ; If (Z != 1) then goto LABEL3 (at this point always true)

LABEL2:
      MOV A,B     ; A = B

LABEL1:
      CMA         ; A = A'
      STA 3000H   ; M(3000H) = A
      JMP START   ; This is the change: Run forever
      RST 1       ; Terminate program

END               ; End of program

; Explanation:
; Return least significant 1's position
; e.g. xxxx1000 -> 100
; e.g. 10000000 -> 1000
; e.g. xxxxxxx1 -> 1
```

Άσκηση 2

Ξεκινήσαμε την επίλυση της συγκεκριμένης άσκησης παράλληλα, πριν σχηματιστεί η ομάδα μας, οπότε παραθέτουμε τους 2 κώδικες Assembly (αρχεία `assignment1_ex2_el19830.8085` και `assignment1_ex2_el20435.8085`) της άσκησης μαζί με σχόλια επεξήγησης της λειτουργίας των:

```
IN 10H
LXI B,01F4H           ; delay
MVI A,FEH             ; LED starting position = 11111110
MOV E,A               ; save iinitial led output 11111110
STA 3000H             ; show initial  led output 11111110

INPUT:
      CALL DELB
      LDA 2000H       ; take input
      ANI 03H         ; check if lsb and 2nd lsb are on
      CPI 01H
      JZ LSB_ON_L     ; if 01 goto LSB_ON_L
      CPI 00H
      JZ LSB_OFF      ; else if 00 goto LSB_OFF
      JMP INPUT       ; in any other case, 2nd lsb is on so stop

LSB_ON_L:
      MOV A,E         ; restore led output
      CPI 7FH         ; check if output is 01111111
      JZ LSB_ON_R     ; if so move right
      RLC             ; else rlc
      MOV E,A         ; save led output
      STA 3000H
      JMP INPUT

LSB_ON_R:
      MOV A,E         ; restore led output
      CPI FEH         ; check if output is 11111110
      JZ LSB_ON_L     ; if so move left
      RRC             ; else rrc
      MOV E,A         ; save led output
      STA 3000H
      INPUT_HELP:
            CALL DELB
            LDA 2000H           ; take input
            ANI 03H             ; check if lsb and 2nd lsb are on
            CPI 01H
            JZ LSB_ON_R         ; if 01 move right
            CPI 00H
            JZ LSB_OFF          ; else if 00 goto LSB_OFF
            JMP INPUT_HELP      ; in other case, 2nd lsb is on so stop

LSB_OFF:
      MOV A,E         ; restore led output
      RLC             ; rotate left
      MOV E,A         ; save new value of led output after rlc
      STA 3000H
      JMP INPUT

END
```

Και:

```
IN 10H
LXI B,02F4H                ; Delay
MVI A,FEH                  ; LED starting position = 11111110
MVI D,00H                  ; (D = 0) => go left, (D = 1) => go right
MOV E,A                    ; save initial led output 11111110
STA 3000H                  ; show initial  led output 11111110

LOOP_FOR_INPUT:
     CALL DELB
     LDA 2000H             ; take input
     CPI 02H               ; A = 10 => halt
     JZ LOOP_FOR_INPUT
     CPI 03H               ; A = 11 => halt
     JZ LOOP_FOR_INPUT
     CPI 00H               ; A = 00 => cycle left
     JZ CYCLE_LEFT
     JMP OSCILLATE         ; Else => oscillate

CYCLE_LEFT:
     MVI D,00H             ; Next oscillation will go left
     MOV A,E               ; Retrieve LED's current position
     RLC                   ; Cycle left output
     MOV E,A               ; save current LED output
     STA 3000H             ; Show output
     JMP LOOP_FOR_INPUT

CYCLE_RIGHT:
     MVI D,01H
     MOV A,E               ; Retrieve LED's current position
     RRC                   ; Cycle right output
     MOV E,A               ; save current LED output
     STA 3000H             ; Show output
     JMP LOOP_FOR_INPUT

OSCILLATE:
     MOV A,E               ; Retrieve output
     CPI 7FH               ; A = 01111111 => change direction
     JZ CHANGE_DIRECTION
     CPI FEH               ; A = 11111110 => change direction
     JZ CYCLE_LEFT         ; CHANGE_DIRECTION to left
     OPPOSITE_DIRECTION:
          MOV A,D
          CPI 00H          ; Select direction
          JZ CYCLE_LEFT
          JMP CYCLE_RIGHT

CHANGE_DIRECTION:
     MOV A,D
     XRI 01H                    ; 00 XOR 01 = 01, 01 XOR 01 = 00
     MOV D,A                    ; D toggles between 00 and 01
     JMP OPPOSITE_DIRECTION     ; So direction changes

END
```

Άσκηση 3

Παραθέτουμε τον κώδικα Assembly της άσκησης (αρχείο `assignment1_ex3.8085`) μαζί με σχόλια επεξήγησης της λειτουργίας του:

```
IN 10H
LXI B,01F4H           ; delay for "blinking"

START:
      LDA 2000H
      MVI E,FFH       ; E = -1 (2's complement)
      CPI 63H         ; check if number > 99
      JC DECA         ; if not, goto deca
      CPI C7H         ; if number > 99 test if number > 199
      JNC GR_199
      SUI 64H         ; if 99 < number < 200 remove 100

DECA:
      INR E
      SUI 0AH         ; remove 10 from input
      JNC DECA        ; if input > 0 continue subbing
      ADI 0AH         ; else restore negative remainder
      MOV D,A         ; save units
      MOV A,E         ; restore ten's
      RRC
      RRC
      RRC
      RRC             ; bring ten's to first 4 msbs with 4 rrc
      ADD D           ; 4 msbs contain ten's, 4 lsbs contain units
      CMA
      STA 3000H
      JMP START

GR_199:
      MVI A,F0H       ; A = 11110000
      STA 3000H       ; show A
      CALL DELB       ; wait
      MVI A,FFH       ; A = 11111111 (for "blinking")
      STA 3000H       ; show A
      CALL DELB       ; wait
      JMP START
END
```

Λειτουργία του προγράμματος

Η λειτουργία του προγράμματος είναι η εξής:

Αρχικά εκχωρούμε το input που δίνει ο χρήστεις από τους διακόπτες στον καταχωρητή A.

1. Αν $A \leq 99$ πηγαίνουμε στο label DECA όπου αφαιρούμε δεκάδες έως ότου ο A να έχει τιμή μικρότερη του 10. Τότε τυπώνουμε στα 4 σημαντικότερα LEDs τις δεκάδες (με 4 δεξιόστροφες περιστροφές) και στα 4 λιγότερο σημαντικά τις μονάδες.
2. Αν $100 \leq A < 200$ αφαιρούμε 100 και εκτελούμε τη διαδικασία στο label DECA, όπως και στο (1)
3. Αν $A \geq 200$ πηγαίνουμε στο label GR_199 όπου αναβοσβήνουμε τα 4 λιγότερο σημαντικά LEDs

Βασιλείου Δημήτριος A.M.: el19830
Ζαρίφης Στυλιανός A.M.: el20435
19/04/2022

Άσκηση 4

Για κάθε μια τεχνολογία έχουμε ότι:

$$\text{Κόστος} = \text{Αρχικό} + (\text{Κόστος} - \text{ICs} + \text{Κόστος} - \text{κατασκευής}) \cdot \text{Πλήθος τεμαχίων}$$

Τεχνολογία 1

Κόστος για n τεμάχια: $\overline{K}_1[n] = 15000 + (10 + 10)n$

Άρα κόστος ανά τεμάχιο: $K_1[n] = \overline{K}_1[n]/n = 20 + 15000/n$

Τεχνολογία 2

Κόστος για n τεμάχια: $\overline{K}_2[n] = 7000 + (50 + 10)n$

Άρα κόστος ανά τεμάχιο: $K_2[n] = \overline{K}_2[n]/n = 60 + 7000/n$

Τεχνολογία 3

Κόστος για n τεμάχια: $\overline{K}_3[n] = 47000 + (2 + 2)n$

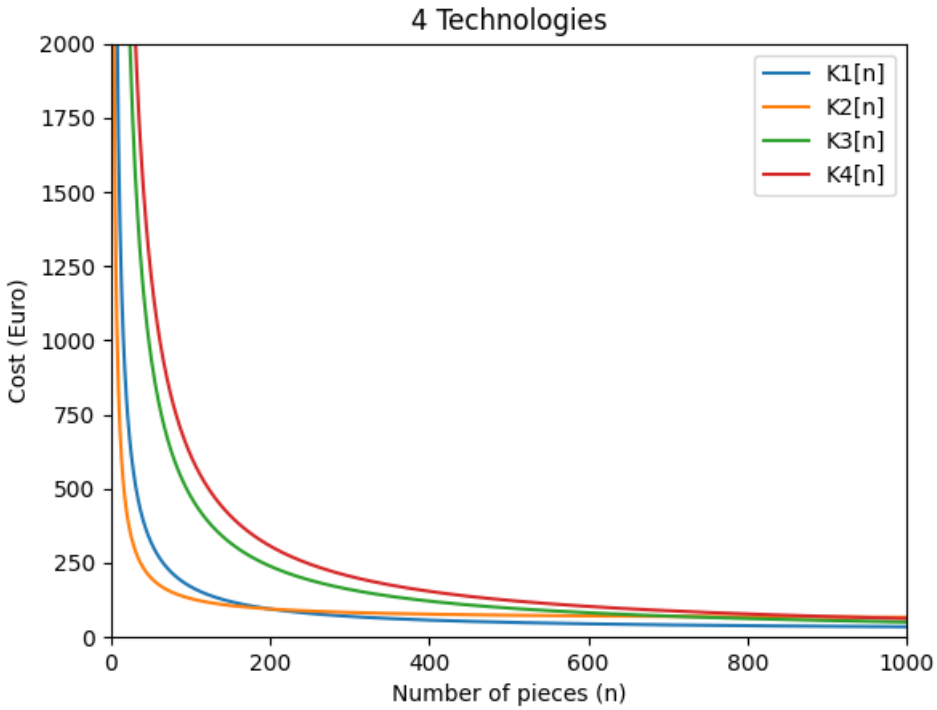
Άρα κόστος ανά τεμάχιο: $K_2[n] = \overline{K}_2[n]/n = 4 + 47000/n$

Τεχνολογία 4

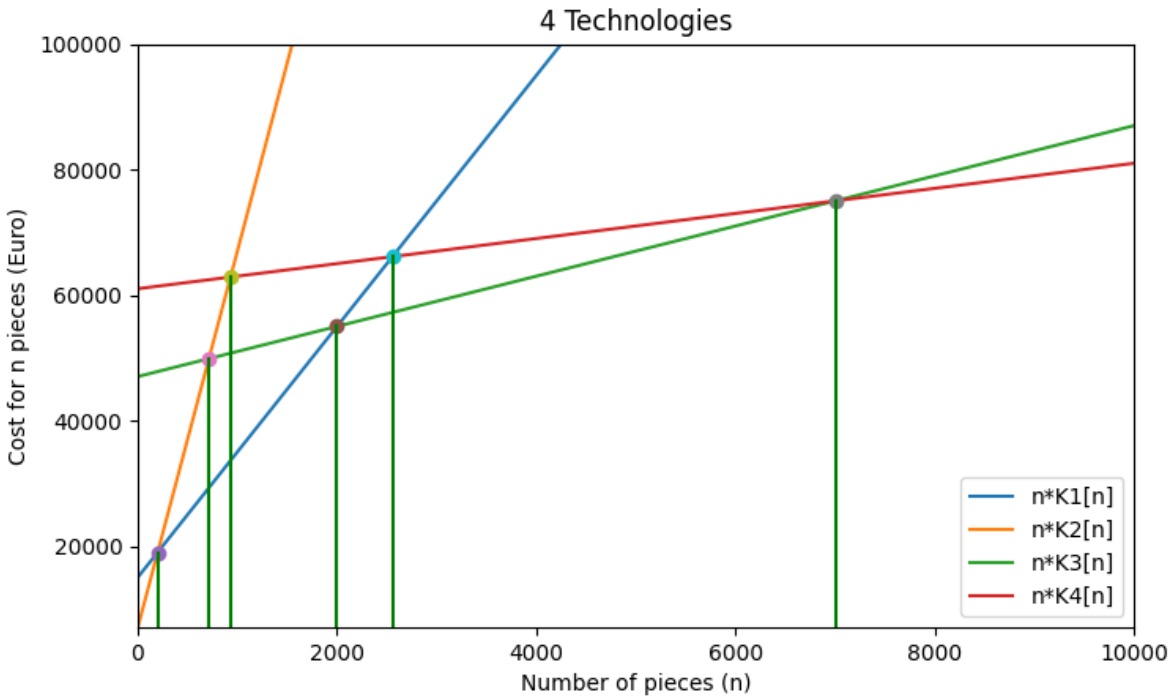
Κόστος για n τεμάχια: $\overline{K}_4[n] = 61000 + (1 + 1)n$

Άρα κόστος ανά τεμάχιο: $K_4[n] = \overline{K}_4[n]/n = 2 + 61000/n$

Με ένα απλό πρόγραμμα σε Python σχεδιάζουμε τις παραπάνω καμπύλες (ενώνοντας τις διαδοχικές διακριτές τιμές για κάθε μία τεχνολογία για καλύτερη εποπτεία) και έχουμε:



Παραθέτουμε ένα ακόμα γράφημα με το κόστος για n τεμάχια της κάθε τεχνολογίας (των συναρτήσεων $\overline{K}_1[n], \overline{K}_2[n], \overline{K}_3[n], \overline{K}_4[n]$ που υπολογίσαμε στην αρχή, δηλαδή):



Εξισώνοντας τις $\overline{K}_i[n], i = 1, 2, 3, 4$ ανά ζεύγη, βρήκαμε το πλήθος των τεμαχίων για το οποίο τέμνονται:

1. $\overline{K}_1[n] = \overline{K}_2[n]$ για $n = n_1 = 200$
2. $\overline{K}_1[n] = \overline{K}_3[n]$ για $n = n_2 = 2000$
3. $\overline{K}_1[n] = \overline{K}_4[n]$ για $n = n_3 \cong 2555$
4. $\overline{K}_2[n] = \overline{K}_3[n]$ για $n = n_4 \cong 714$
5. $\overline{K}_2[n] = \overline{K}_4[n]$ για $n = n_5 \cong 931$
6. $\overline{K}_3[n] = \overline{K}_4[n]$ για $n = n_6 = 7000$

Οπότε λύνοντας αντίστοιχες ανισώσεις ή παρατηρώντας το τελευταίο διάγραμμα και βρίσκοντας το χαμηλότερο κόστος για ορισμένο διάστημα τεμαχίων, μπορούμε να βρούμε τις εξής συμφερότερες περιοχές για κάθε τεχνολογία:

1. Τεχνολογία 1: Συμφέρει για 200 ως 2000 τεμάχια
2. Τεχνολογία 2: Συμφέρει για 1 ως 200 τεμάχια
3. Τεχνολογία 3: Συμφέρει για 2000 ως 7000 τεμάχια
4. Τεχνολογία 4: Συμφέρει για 7000 και περισσότερα τεμάχια

Έστω ότι η τεχνολογία των FPGAs κοστίζει c $€/τεμάχιο$ αντί για 50. Θα εξαφανίσει την επιλογή της 1ης αν για το εύρος τεμαχίων που συμφέρει η τεχνολογία 1, είναι πιο φθηνή η τεχνολογία 2.

Δηλαδή αν:

$$\bar{K}_1[n] \geq \bar{K}_2[n] \Rightarrow 15000 + (10+10)n \geq 7000 + (c+10)n \Rightarrow 8000 \geq (c-10)n \Rightarrow \boldsymbol{c \leq 10 + 8000/n}$$

Όποτε, επιθυμούμε να ισχύει η ανισότητα για κάθε n. Η $10 + 8000/n$ ως φθίνουσα έχει ασυμπτωτική ελάχιστη τιμή 10 για $n \to +\infty$. Οπότε, πρέπει η c να είναι μικρότερη από την ελάχιστη αυτή τιμή, άρα αρκεί $\boldsymbol{0 < c \leq 10}$, όπου c το κόστος ανά τεμάχιο των ICs.