Συστήματα Μικροϋπολογιστών

Μιχάλης Παπαδόπουλλος (031 14702)

Χαράλαμπος Κάρδαρης (031 * * * *)

(1η σειρά ασκήσεων)

Άσκηση 1: Μας ζητείται να αποκωδικοποιήσουμε το ακόλουθο κομμάτι κώδικα που μας δίνεται σε γλώσσα μηχανής (Αρχιτεκτονική Intel 8085)

0E 08 **3A** 00 20 **17 DA** 0D 08 **OD C2** 05 08 **79 2F 32** 00 30 **CF**

```
; Michalis Papadopoullos
; 031 14702
; Exercise 1 - Microsystems
; Intel 8085 Disassembly process

jmp start ; jump to the program
; data section

; ------------
; code section
; start = 0x0800
start:  mvi C, 08 ; labels must be followed by an instruction
        lda 2000H ; load ACCUMULATOR = MEM[0x2000]
        ral       ; rotate ACCUMULATOR left
        jc  080DH ; jump if CARRY flag is set
        dcr C     ; decrement C
        jnz 0805H ; jump if not flag ZERO is set to 1
        mov A, C  ; move contents of C to A
        cma       ; complement ACCUMULATOR: A = ~A
        sta 3000H ; store contents of A at address 0x3000
        RST 1     ; RESET
        hlt       ; halt execution
; ---- [ EOF ] --
```

Σημείωση: Στον πιο πάνω κώδικα, δεν έδωσα κάποιο ***LABEL*** για τις διευθύνσεις αναφοράς **0x0805** και **0x080D.** Για να τρέξω την προσομοίωση ονόμασα αντίστοιχα ***L1:*** και ***L2:*** τις διευθύνσεις αυτές πριν απο την αρχή της εντολής και αντικατέστησα τις διευθύνσεις στις εντολές ***jump*** με το αντίστοιχο όνομα.

```
ADDR    OPCODE        MNEMONICS
---------------------------------
0800 |   0E     |    ; MVI C, 08
0801 |   08     |
0802 |   3A     |    ; LDA 2000
0803 |   00     |
0804 |   20     |
0805 |   17     |    ; RAL
0806 |   DA     |    ; JC 080D
0807 |   0D     |
0808 |   08     |
0809 |   0D     |    ; DCR C
080A |   C2     |    ; JNZ 0805
080B |   05     |
080C |   08     |
080D |   79     |    ; MOV A, C
080E |   2F     |    ; CMA
080F |   32     |    ; STA 3000
0810 |   00     |
0811 |   30     |
0812 |   CF     |    ; RST 1
---------------------------------
```

Δίπλα, δίνεται το πρόγραμμα σε Assembly, μαζί με τις διευθύνσεις κάθε byte στη μνήμη.

Η αποκωδικοποίηση των εντολών, έγινε με βάση τον πίνακα 2 – παράρτημα 2 των σημειώσεων που μας δίνονταν.

Για να εκτελείται ο κώδικας αυτός συνέχεια (infinite loop), αρκεί να αντικαταστήσουμε την εντολή **RST *(reset)*** με ένα ***unconditional jump*** στην αρχή του κώδικα. Δηλαδή: **jmp start** όπου ***start*** είναι ένα ***LABEL*** στην αρχή του κώδικα, με διεύθυνση ***0x0800***.

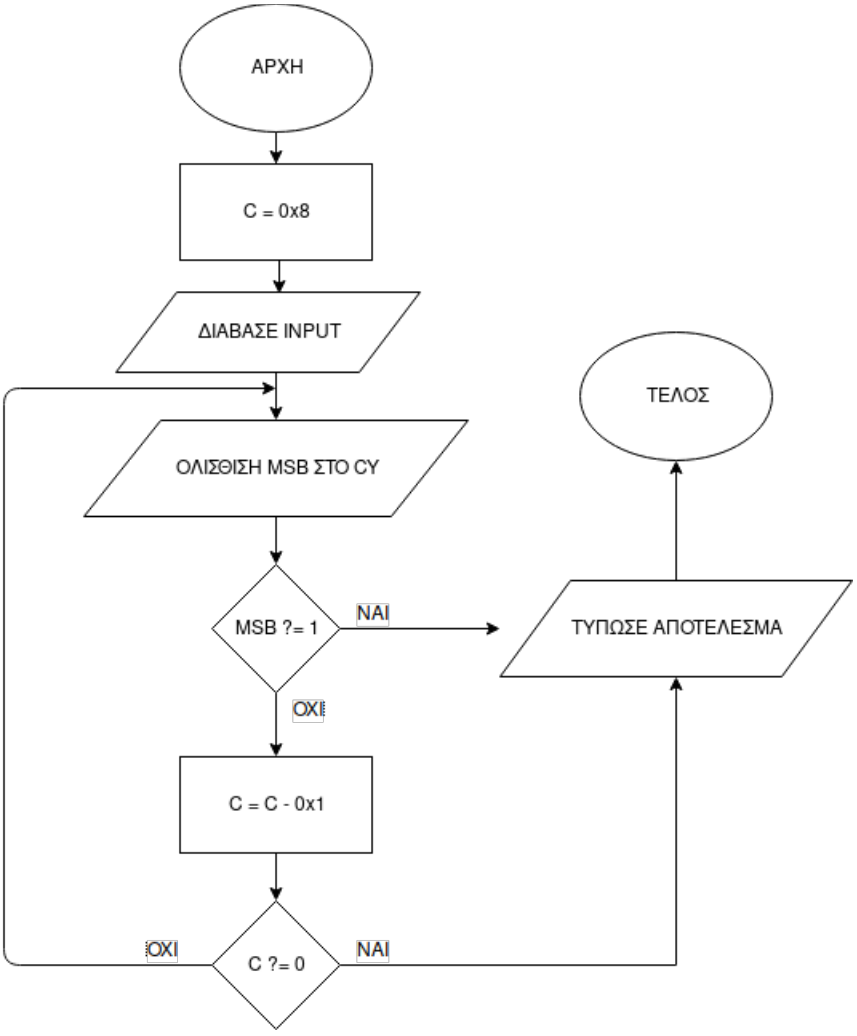
```
; Michalis Papadopoullos
; 031 14702
; Exercise 1 - Microsystems
; Intel 8085 Disassembly process

jmp start ; jump to the program
; data section

; ------------
; code section
; start = 0x0800
start:  mvi C, 08 ; labels must be followed by an instruction
        lda 2000  ; load ACCUMULATOR = MEM[0x2000] (INPUT PORT)
L1:     ral       ; rotate ACCUMULATOR left
        jc  L2    ; jump @L2 if CARRY flag is set
        dcr C     ; decrement C
        jnz L1    ; jump @L1 if not flag ZERO is set to 1
L2:     mov A, C  ; move contents of C to A
        cma       ; complement ACCUMULATOR: A = ~A
        sta 3000  ; store contents of A at address 0x3000 (OUTPUT PORT)
        jmp start ; unconditional jump @start
        hlt       ; halt execution
; ---- [ EOF ] --
```

Τρέχοντας τον παραπάνω κώδικα στον προσομοιωτή ***GNUSim8085*** μπορούμε να καταλάβουμε κάποια πράγματα σχετικά με τον κώδικα. Αρχικά, φορτώνει την αρχική τιμή ***0x08*** στον καταχωρητή ***C*** και διαβάζει δεδομένα από την θύρα εισόδου (***lda 0x2000***).

> ***RAL:*** Each bit of the ***accumulator*** is rotated left by one position through the ***Carry Flag***. Bit D7 is placed in the ***Carry Flag***, and the ***Carry Flag*** is placed in the **least significant position** D0. **CY** (*carry bit*) is modified according to bit D7.

Το ***μLab*** έχει μια θύρα εισόδου (8 γραμμών) και μια θύρα εξόδου που αποτελείται απο 8 ***LED*** – μια για κάθε γραμμή εξόδου. Όταν μια γραμμή εξόδου είναι ***HIGH*** τότε το αντίστοιχο *LED* στην θύρα εξόδου είναι σβηστό. Γι' αυτό το λόγο προσθέτουμε την εντολή ***CMA*** η οποία, μέσω της **ALU** δίνει το λογικό συμπλήρωμα ως προς 1 του 8bit αριθμού εισόδου – κάνοντας έτσι τα αντίστοιχα *LED* της γραμμής εξόδου να είναι αναμμένα όταν βρίσκεται σε κατάσταση *HIGH*.

Το διάγραμμα ροής του προγράμματος φαίνεται πιο κάτω:



– Σχεδιάστηκε με τη βοήθεια της πλατφόρμας draw.io