EASTERN MEDITERRANEAN UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

CMPE 323: Microprocessors

Final Exam

Date: 12 / 01 / 2018

Lecturer: Hasan Kömürcügil **Time Allowed:** 120 minutes

Name and Surname: SOLUTION

Student Number:.......................................

- There are **5** questions in this exam paper.
- Answer **all** questions.
- Write **clearly** and **tidily.**
- Correct answers **without sufficient explanation** might not get full points!
- Mobile phones **must be switched off** in the exam room.

Question	Points Gained
Q1 *(27 points)*	
Q2 *(20 points)*	
Q3 *(16 points)*	
Q4 *(23 points)*	
Q5 *(14 points)*	
Total	

Q1) [27 *points*]

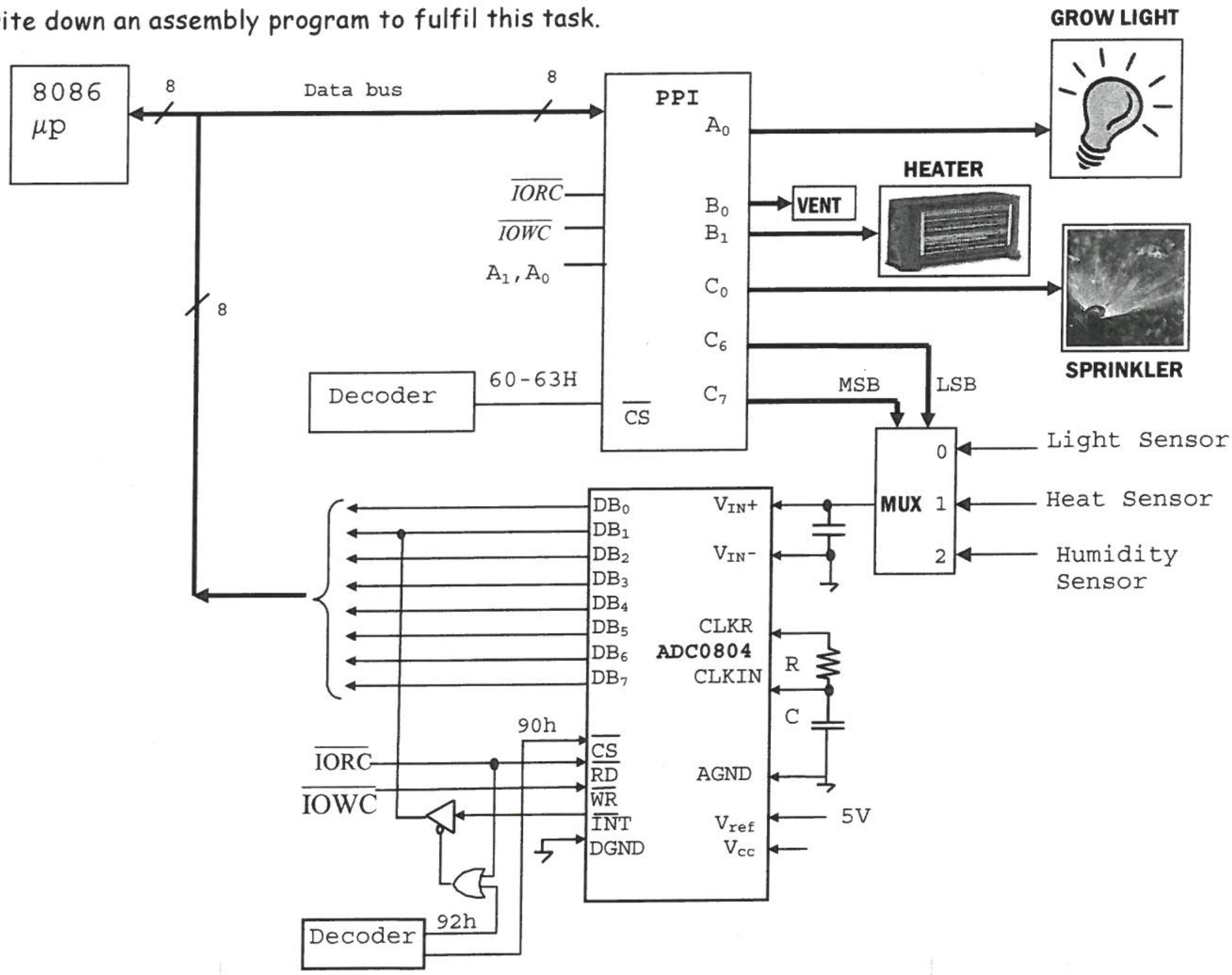
It is required to design the following microprocessor controlled greenhouse to provide the optimum conditions for growing plants. The system works as follows. The ADC gets data from the sensors (light, heat and humidity) through a multiplexer (MUX). The digital output of ADC is used to turn ON/OFF the grow light, heater and sprinkler and to CLOSE/OPEN the vent according to the following conditions.

1) (a) If there is **enough light**, the **grow light** is turned OFF (A_0=0)
   (b) If it is **too dark**, the **grow light** is turned ON (A_0=1)
2) (a) If it is **too hot**, the **heater** is turned OFF (B_1=0) and **vent** is OPENED (B_0=1)
   (b) If it is **too cold**, the **heater** is turned ON (B_1=1) and **vent** is CLOSED (B_0=0)
3) (a) If the **soil is wet**, the **sprinkler** is turned ~~ON~~ OFF (C_0=1) $C_0=0$
   (b) If the **soil is dry**, the **sprinkler** is turned ~~OFF~~ ON (C_0=0) $C_0=1$

The digitized values of data received from the sensors are given in the following table.

Enough Light	Too Dark	Too Hot	Too Cold	Wet Soil	Dry Soil
10000000	00001000	01000000	00000100	00100000	00000010

Write down an assembly program to fulfil this task.



Note1: The command register format of the PPI is: 1 0 0 A Ch 0 B Cl — 1=Input, 0=Output

Note 2: The 7-segmet display format is :

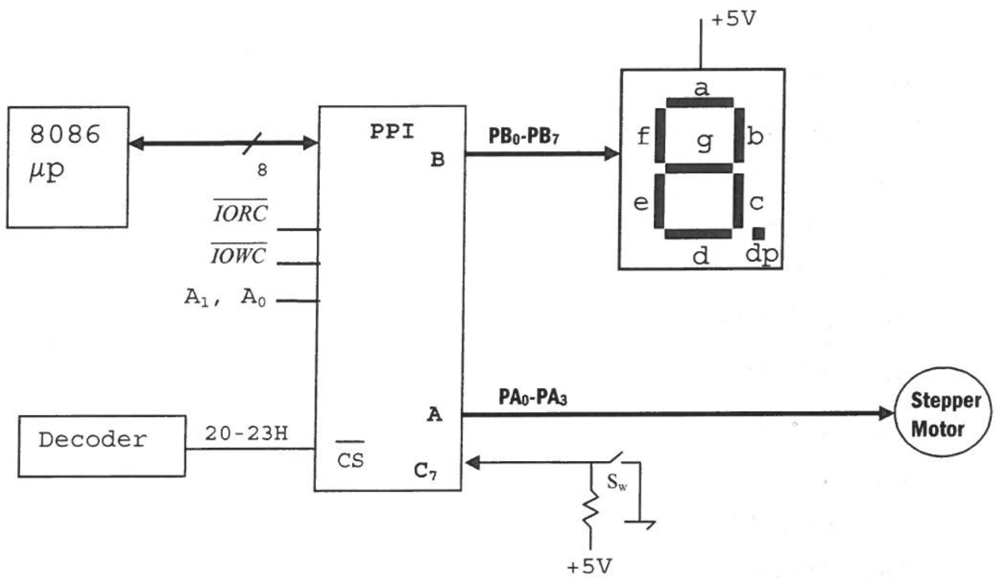
A7	A6	A5	A4	A3	A2	A1	A0
a	b	c	d	e	f	g	dp

```
            .Code
            mov al, 10000000b
            out 63h, al
            mov bl, 1
            mov al, 0       ; select Light Sensor
            out 62h, al
Start:      out 90h, al
Notrdy:     in al, 92h
            test al, 02h
            jnz Notrdy
            in al, 90h
            cmp bl, 1
            je GrowL
            cmp bl, 2
            je Heater
            cmp bl, 3
            je Sprinkler
GrowL:      cmp al, 80h
            Jb Dark
Enough:     mov al, 0       ; A0 = 0
            out 60h, al
            jmp cont1
Dark:       mov al, 1       ; A0 = 1
            out 60h, al
Cont1:      mov bl, 2
            mov al, 40h ; Select Heat Sensor
            out 62h, al
            jmp start
Heater:     cmp al, 40h
            Jb Cold
Hot:        mov al, 1       ; B1=0 & B0=1
            out 61h, al
            jmp cont2
Cold:       mov al, 2       ; B1=1 & B0=0
            out 61h, al
Cont2:      mov bl, 3
            mov al, 80h; Select H.Sensor
            out 62h, al
            jmp start

Sprinkler:  cmp al, 20h
            jb DryS
WetS:       mov al, 0       ; C0=0
            out 62h, al
            jmp cont3
DryS:       mov al, 1       ; C0=1
            out 62h, al
Cont3:      mov bl, 1
            Mov al, 0
            out 62h, al
            jmp start
```

Q2) [20 *points*]

In the following system, the stepper motor is rotated 5 turns in the clockwise direction (to the right) and "5" is displayed on the 7-segment display unit when the switch (S_w) is open. The motor rotates 8 rotations in the anti-clockwise direction (to the left) and "8" is displayed on the 7-segment display unit when the switch is closed. Once the stepper motor direction is determined, it is assumed that the position of switch is not changed until the motor completes its desired rotations. When the motor completes its rotation, the program should check the switch position to determine the new direction of rotation. Complete the following assembly program that fulfils this task.



Note1: The command register format of the PPI is: `1 0 0 A Ch 0 B Cl` (1=Input, 0=Output)

Note 2: The 7-segmet display format is :

A7	A6	A5	A4	A3	A2	A1	A0
a	b	c	d	e	f	g	dp

```
        Dosseg
        .Model small
        .Code
        Mov al, 10001000B
        Out 23h,al
        Mov bl,11001100B
Start:  In al, 22h
        And al,10000000B
        Shl al, 1
        Jnc  Left
Right:  Mov al, 49h
        Out 21h,al
        Mov cx, 5
RotR:   Ror bl,1
        Mov al,bl
        Out 20h,al
        Call Delay
        Loop RotR
        Mov bl, al
        Jmp Start
```

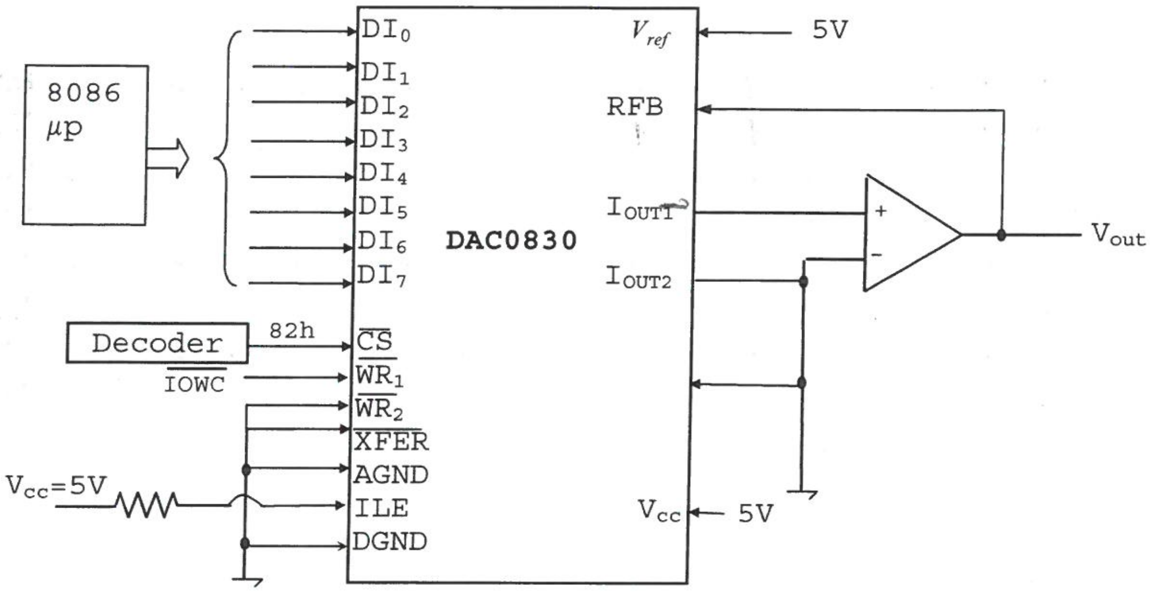
```
Left:   Mov al,01h
        Out 21h, al
        Mov cx,8
RotL:   Rol bl, 1
        Mov al,bl
        Out 20h,al
        Call Delay
        Loop RotL
        Mov bl, al
        Jmp Start

Delay:  ..................  1μs
        ret
```

Q3) [16 *points*]

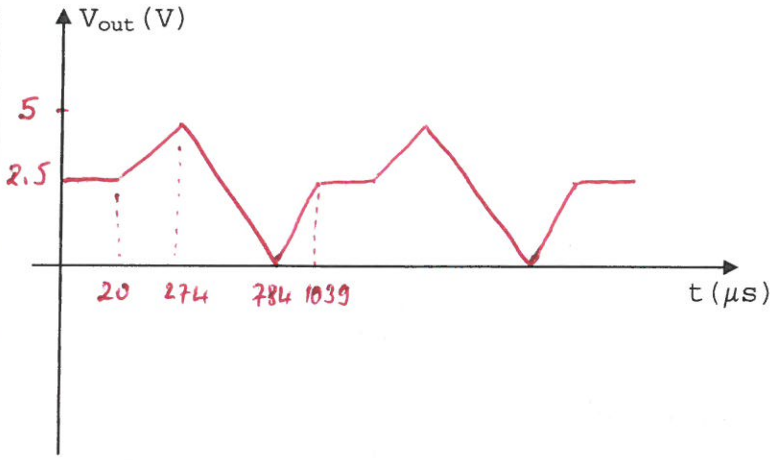
Consider the following DAC0830 interfaced to an 8086 microprocessor. An assembly program which is written to produce an analog waveform via DAC at V_{out} is provided below.

(a) *[12 points]* Assuming that the hexadecimal numbers 00h and FFh correspond to 0V and 5V, respectively, trace the assembly program and draw V_{out} for two periods. Show voltage levels and time on the waveform in detail.




```
            Dosseg
            .Model small
            .Code
            Mov al,127
Start:      Mov cx,10
Again:      Out 82h,al
            Call delay
            Loop Again
Next:       Inc al
            Out 82h,al
            Call delay
            Cmp al,255
            jb Next
Continue:   Dec al
            Out 82h,al
            Call Delay
            Cmp al,0
            Ja Continue
Repeat:     Inc al
            Out 82h, al
            Call Delay
            Cmp al,127
            Jb Repeat
            Jmp Start

Delay:      ..................  2μs
            ret
```



(b) *[4 points]* Find the approximate period and frequency of V_{out}.

$T = 1039 \mu s$

$f = \frac{1}{T} = \frac{1}{1039} \times 10^6$ Hz

Q4) [23 pts]

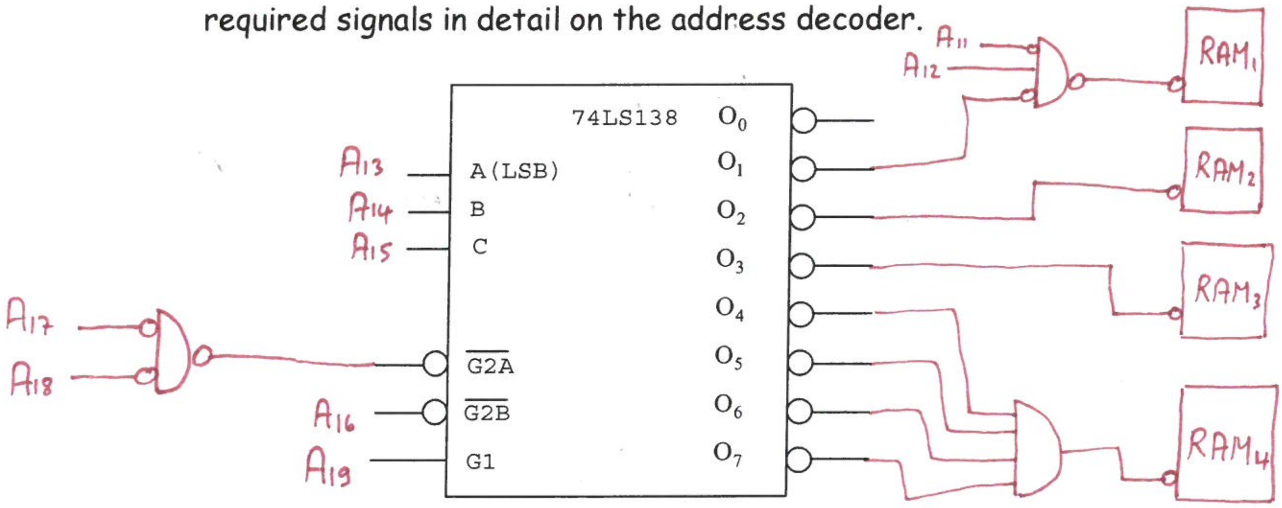
It is required to interface the following memory chips to an 80386 microprocessor based system which has 20-bit address and 8-bit data buses in the following address ranges:

- 1 (2Kx8) ROM chip to decode 83000h-837FFh
- 2 (8Kx8) ROM chips to decode 84000h-87FFFh
- 1 (32Kx8) RAM chip to decode 88000h-8FFFFh

a) *[4 points]* Fill in the following table

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Range
ϕ	0	0	0	0	0	1	ϕ	0	0	0	0	0	0	0	0	0	0	0	0	83000
ϕ	0	0	0	0	0	1	ϕ	0	ϕ	1	1	1	1	1	1	1	1	1	1	837FF
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	84000
1	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	85FFF
1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	86000
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	87FFF
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88000
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	8FFFF

b) *[16 points]* Using 74LS138 decoder shown below, design an <u>address decoding circuit</u> to decode the above address ranges. Show your connections and the required signals in detail on the address decoder.



c) *[3 points]* How much memory (both size and its range) is available in the memory map for an additional memory chip?

Available ranges: 00000 – 82FFF

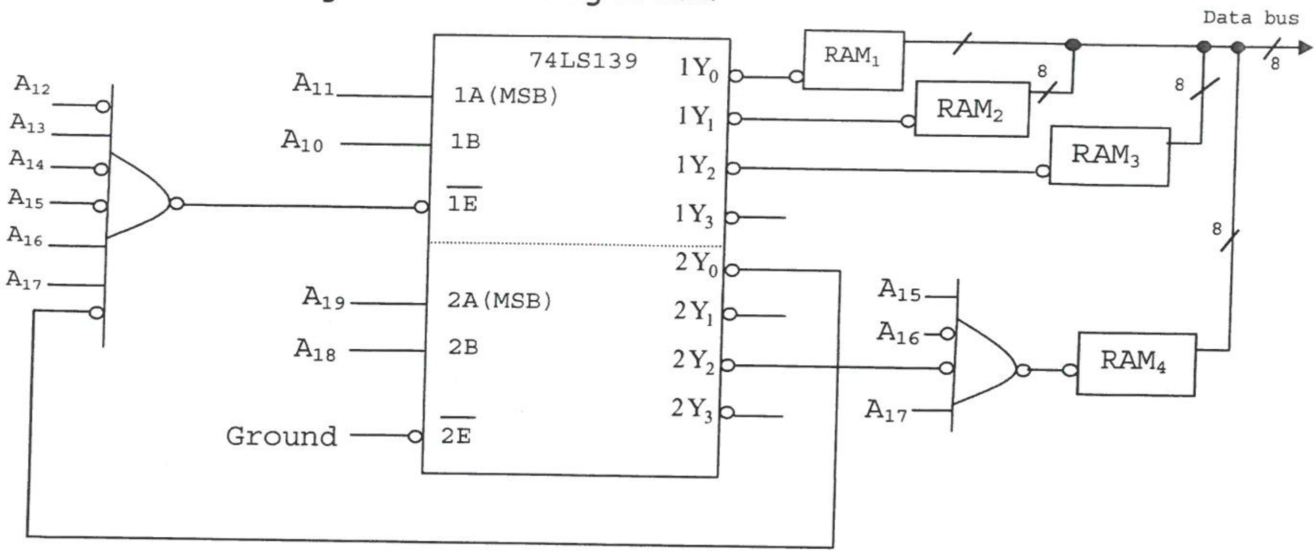
83800 – 83FFF

90000 – FFFFF

Available size: 1024KB – (2+8+8+32) = 974KB

Q5) [14 *points*]

Consider the following address decoding circuit.



a) *[8 points]* Determine the decoded address range (in Hexadecimal) and size (in KB) for the RAM chips and record them into the following table.

	Decoded address range (in hex)	Decoded size (in KB)
RAM_1	32000 - 323FF	1
RAM_2	32400 - 327FF	1
RAM_3	32800 - 32BFF	1
RAM_4	A8000 - AFFFF	32

b) *[6 points]* It is required to interface 3 additional RAM chips to the existing outputs ($1Y_3$, $2Y_1$ and $2Y_3$) of the address decoding circuit shown above. Find the possible address range and size of each RAM chip and record them into the following table.

	Decoded address range (in hex)	Decoded size (in MB)
$1Y_3$ (RAM_5)	32C000 - 32FFF	0.016
$2Y_1$ (RAM_6)	40000 - 7FFFF	0.256
$2Y_3$ (RAM_7)	C0000 - FFFFF	0.256