Union College
ECE352/552 Midterm Exam
Thursday May 1st, 2014

Name: Solution

Possible points	Actual Points
Problem 1: 15 points	
Problem 2: 20 points	
Problem 3: 30 points	
Problem 4: 10 points	
Problem 5: 10 points	
Problem 6: 15 points	
TOTAL (out of 100)	

This exam is a closed book and closed notes exam. You will be provided with the C8051F020 datasheet. You have the full class period to complete the exam. Show work whenever possible to receive partial ctedit.

When asked to write code you can assume that the c8051F020.inc file, for assembler, and the file c8051f020.h is available. You do not need to write the include expression for this.

Problem 1 (15 points):

Write down the final values in the register **A** in the space provided after each instruction has executed.

```
$include (c8051F020.inc)

                cseg at 0h
                ljmp Main

                cseg at 100h
Main:           mov  WDTCN, #0DEh
                mov  WDTCN, #0ADh

                mov a, #0xB9
                cpl a               ; A = _46__
                mov R0, #0x04
                orl a, #0xF0        ; A = _F6__
                rr  a               ; A = _7B_
                anl a, #0xF0        ; A = _70__
                add a, R0           ; A = _74__
                swap a              ; A = _47__

                nop

end
```

Problem 2 (20 points):

Write down the final values in all memory locations modified in this code. Instead of writing down the SFR or register name, enter the address in memory of that location. Use the "box" method to analyze the probram.

```
$include (c8051F020.inc)

cseg at 0h
        ljmp Main

cseg at 100h
Main:
        mov  WDTCN, #0DEh
        mov  WDTCN, #0ADh

        mov 0x51, #51h
        mov 0x50, 0x51
        mov a, #01
        mov R0, #0x50
        mov DPTR, #table
        movc a, @a + DPTR
        mov  @R0, a
        nop

cseg at 20FFh
table: db 0Ah, 0Bh, 1Ch

end
```

Ram

0x51	51
0x50	0B

A = 0x0E0

0B

R0 = 0x00

0x50

DPTR = 82 4

20FF

01 20FF = 2100

50

Code

2101	1C
2100	0B
20FF	0A

Final values

0x50 = 0B; 0x51 = 51; R0 = 0x50; A = 0B;

DPTR = 20FF; 20FF = 0A; 2100 = 0D; 2101 = 1C

Problem 3 (30 points):

Numbers, letters, punctuation, etc., are represented in computers using ASCII codes. The numbers 0 to 9 are represented by 8-bit ASCII characters with numeric values from 30 HEX to 39 HEX. Thus, the number 3, 0000 0011, in a memory location can be converted to its ASCII representation by adding 30 HEX to the numeric value (0011 0011 - 32 HEX).

a. Write an algorithm for a C function to convert an unsigned char value that is passed to the function, in a parameter value containing a numeric value from 0 to 9, to an ASCII character.

Function convert

Create an parameter variable of type unsigned char called number
Create a function local variable of type unsigned char called ascii_value
Ascii_value = number + 0x30
Return (ascii_value

b. Write the C code to implement the ASCII convert subroutine. Complete the function below.

```
unsigned char convert(unsigned char number) {

unsigned char ascii_value;
    ascii_value = number + 0x30;
    return(ascii_value);
}
```

c. Rewrite the C code from part B in assembler to implement the ASCII convert subroutine. Complete the subroutine below.
Note: I will pass the number to convert in R1 and return the converted value in R1.

```
Convert:
   push psw
   push acc
   mov a, R1
   add a, #30h
   mov R1, a
   pop acc
   pop psw
ret
```

Problem 4 (10 points):

Calculate the time it will take to execute the following code:

```
mov a, #3       1 cycle
djnz  acc, $    2 cycles
```

a. Using the built in 2 MHz oscillator
T = 1/(2 MHz) = 0.5 us
Number_cycles = (acc)(2) + 1
t = ((2 * 3) + 1) * .5 us = 3.5 us

b. Using the external 22.1184 MHz oscillator
T = 1/ (22.1184 MHz) = 45.211 ns
t = ((2 * 3) + 1) * 45.2 ns = .316 us

Problem 5 (10 points):

Write a C function that will setup port 1 so that pins 0 to 3 are inputs and pins 4 to 7 are outputs. You don't need to configure the crossbar.

```
Void port_init(void) {
   P1MDOUT = 0xF0;
   P1MDIN = 0x0F;
  P1 = 0x0f;
}
```

Problem 6 (15 points):

We are going to use timer 0 in interrupt mode to blink the LED on the development board (connected to port 1 pin 6).

a. Write a C function to initialize timer 0 for 16 bit mode and sysclock to be the internal 2 MHZ oscillator not divided by 12. Include instruction(s) to enable interrupts but you don't need to configure the crossbar.

```
void Timer_Init(void){
    /*
     CKCON -> clock configuration
      TMOD -> mode of timer 0
      TCON -> set timer 0 on
      THO and TL0 count registers - initialize to 0x8000 = 32768(10)
      */
    CKCON = 0x08;  // bit 3 selects timer 0 clock. Set bit to 1 for no / 12
      TMOD = 0x01;  // set mode to 16b
      TL0 = 0x00;  //
      TH0 = 0x80;
      TCON = TCON | 0x10; // enable timer
     LED = 0;
      IE = 0x82;   // Enable TIMER0 interupt enable and global interupts
}
```

b. Write an interrupt service routine (ISR) in C to blink the LED and reinitialize the timer count registers TH0 = 0x80 and TL0 = 0x00.

```
void T0ISR(void) interrupt 1 {
   LED = ~LED;
        TL0 = 0x00;
        TH0 = 0x80;
        TF0 = 0; // reset interupt
}
```

c. Calculate the delay between toggles.

Clock T = 1/ (2 MHz) = .5 us
Start count $8000_{16} = 32768_{10}$
Number_of counts = max counts – start_count = 65536 – 32768 = 32768 counts

Toggle_time = num_counts * T = 32768 * .5 us = 16.4 ms

Note: this is pretty fast and your eyes will only see a flicker.