CSS332: Microcontrollers and Applications
Midterm Mock Exam

curated by The Peanuts

Name. Nonprawich I. ID. 6622772422 .... Section. ..... Seat No. .....

Conditions: Semi-Closed Book

Directions:

1. This exam has 15 pages (including this page).
2. If you finish early, stare at your paper like it's a masterpiece.
3. Write your name clearly, even microcontrollers need proper initialization.
4. Reading the problem is optional but highly recommended.
5. Answers must be written in a structured format. Random noise (scribbles) will not be processed correctly.
6. Going to the toilet may deduct your score, hold it like your grades depend on it.

For solution, click here.

Problem 1

Answer the following questions:

Convert binary 1101 1011 to decimal

219_{10} #

Convert decimal 123 to binary and hexadecimal

$0111\,1011_2$ #

$7B_{16}$ #

Convert hexadecimal 0xA7 to binary and decimal

$1010\,0111_2$ #

167_{10} #

Problem 2

1111 111
1111 0111 +
0001 1001
1 0001 0000

For the addition of 8-bit unsigned numbers: 0xF7 + 0x19, determine:

a) The result in hexadecimal 0x110 #

b) The state of the Carry (C) flag C = 1 #

c) The state of the Zero (Z) flag Z = 0 #

d) The state of the Half-carry (H) flag H = 1 #

101 − 138

For the subtraction of signed 8-bit numbers: 0x65 - 0x8A, determine:

a) The result in hexadecimal 0xDB #

b) The state of the Negative (N) flag The MSB (bit 7) is 1 meaning the result is negative N = 1 #

c) The state of the Overflow (V) flag

d) Is this result correct from a signed arithmetic perspective? Explain.

0110 0101 −
1000 1010 ~ this is already negative
2nd complement
0111 0101 +
1
0111 0110

It goes beyond signed 8-bit range

expected result 101-(-118) = 219

∴ The result is incorrect due to overflow

∴ V = 1 #

⇓

No, overflow occured #

0110 0101 +
0111 0110
1101 1011
D B

Problem 3

Consider the following AVR assembly program:

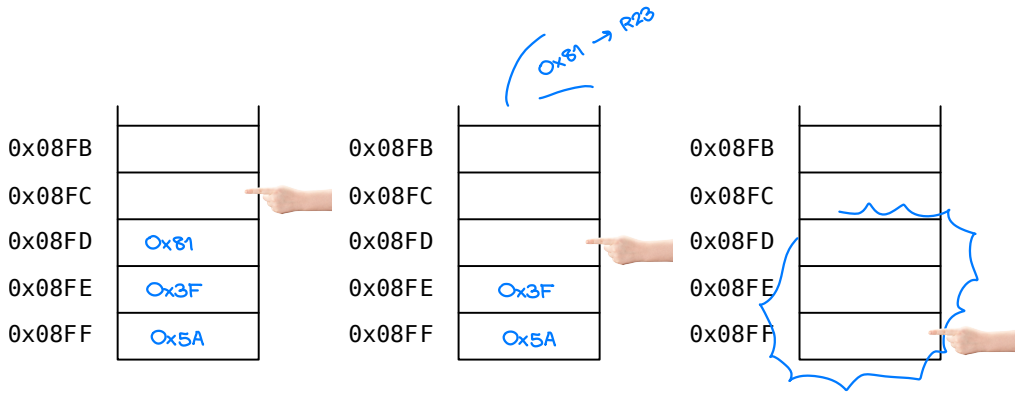
```
.ORG 0
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16

LDI R20, 0x5A
LDI R21, 0x3F
LDI R22, 0x81
PUSH R20
PUSH R21
PUSH R22
POP R23
POP R24
POP R25
```

a) What values will be stored in registers R23, R24, and R25 after executing this program?

R23 = 0x81
R24 = 0x3F #
R25 = 0x5A

ดูจาก stack ข้างล่างได้ !

b) Show the values stored and the stack pointer position each time at *three* points: after all three PUSH instructions, after the first POP instruction, and after all three POP instructions.



c) What would happen if the PUSH and POP instructions were executed without initializing the stack pointer (the first four instructions)? Explain.

```
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16
```

Without initializing the stack pointer, PUSH and POP instructions may cause unpredictable program behavior.

Proper stack initialization is critical for correct stack operations

#

Problem 4

Calculate the total time delay (in milliseconds) for the delay subroutine, assuming the microcontroller is running at 16MHz:

```
DELAY:
    LDI R20, 160
L1:
    LDI R21, 200
L2:
    LDI R22, 250
L3:
    NOP         1
    NOP         1
    DEC R22     1
    BRNE L3     2/1
    DEC R21     1
    BRNE L2     2/1
    DEC R20     1
    BRNE L1     2/1
    RET         4
```

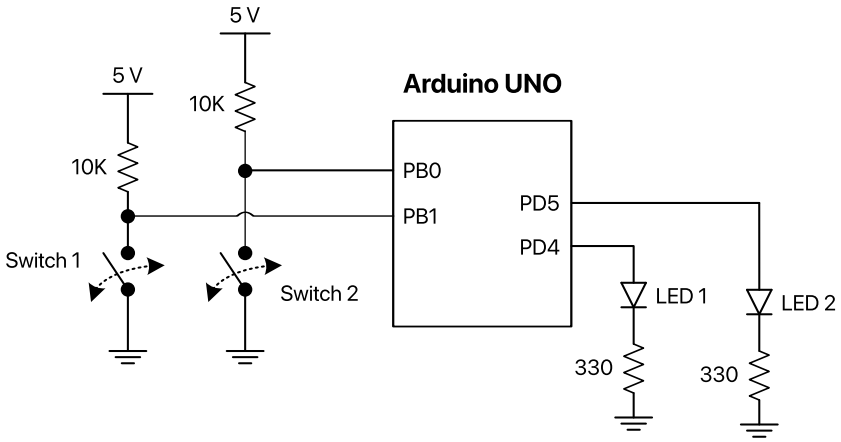
$$1 + \left(\left\{1 + \left[\left\{1 + (5 \times 250) - 1 + 3\right\} \times 200\right] - 1 + 3\right\} \times 160\right) - 1 + 4$$

Show your calculations step by step, counting instruction cycles accurately.

$$\left[1 + \left(\left\{1 + \left[\left\{1 + (5 \times 250) - 1 + 3\right\} \times 200\right] - 1 + 3\right\} \times 160\right) - 1 + 4\right] \times \frac{1}{16M} = 2.506 \text{ seconds } \#$$

Problem 5

The circuit below shows an *Arduino UNO* connected to two switches and two LEDs:



Write a complete assembly program that implements the following functionality:

- Correctly configures the I/O pins (PB0, PB1 as inputs with pull-up resistors; PD4, PD5 as outputs)
- When Switch 1 is pressed (closed state, LOW signal), LED 1 turns ON
- When Switch 1 is not pressed (open state, HIGH signal), LED 1 turns OFF
- Similarly, when Switch 2 is pressed (closed state, LOW signal), LED 2 turns ON
- When Switch 2 is not pressed (open state, HIGH signal), LED 2 turns OFF

This page is intentionally left blank for writing code

```
1           .ORG 0
2
3           ; Configure LED pins as OUTPUT (PD4, PD5)
4           SBI DDRD, 4      ; Set PD4 as output (LED 0)
5           SBI DDRD, 5      ; Set PD5 as output (LED 1)
6
7           ; Turn off LEDs initially
8           CBI PORTD, 4     ; Ensure LED 0 is OFF
9           CBI PORTD, 5     ; Ensure LED 1 is OFF
10
11          ; Configure switch pins as INPUT with pull-up resistors (PB0, PB1)
12          CBI DDRB, 0      ; Set PB0 as input (Switch 0)
13          SBI PORTB, 0     ; Enable pull-up resistor on PB0
14
15          CBI DDRB, 1      ; Set PB1 as input (Switch 1)
16          SBI PORTB, 1     ; Enable pull-up resistor on PB1
17
18  AGAIN:
19          ; Check Switch 0 (PB0)
20          SBIS PINB, 0     ; Skip next instruction if PB0 is HIGH (not pressed)
21          RJMP LED_On0     ; If PB0 is LOW (pressed), jump to LED_On0
22          CBI PORTD, 4     ; Otherwise, turn off LED 0
23
24          ; Check Switch 1 (PB1)
25          SBIS PINB, 1     ; Skip next instruction if PB1 is HIGH (not pressed)
26          RJMP LED_On1     ; If PB1 is LOW (pressed), jump to LED_On1
27          CBI PORTD, 5     ; Otherwise, turn off LED 1
28
29          RJMP AGAIN       ; Repeat loop
30
31  LED_On0:
32          SBI PORTD, 4     ; Turn on LED 0 (PD4)
33          RJMP AGAIN
34
35  LED_On1:
36          SBI PORTD, 5     ; Turn on LED 1 (PD5)
37          RJMP AGAIN
```

Problem 6

Using the 7-segment display pinout provided in the appendix:

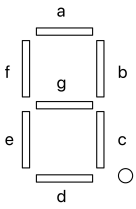
Write a complete assembly program that displays your student ID one digit at a time on a common cathode 7-segment display connected to PORTD.

For example, if your student ID is *6622772422*, the program should display each digit sequentially with an approximately 2-second delay between digits and loop continuously.

(For the delay subroutine, assume the microcontroller is running at **8MHz** and show your calculation of how many cycles your delay loop requires.)

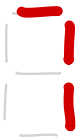
Your student ID has 10 digits. If you store these digits in program memory using the `.DB` directive, explain whether or not the padding effect will occur. Would you need to add an extra byte? Why or why not?

Padding effect will not happen because our student ID is 10 digits long, and it's even number, so it's going to be okay since .DB defines bytes in program memory



g f e d c b a

g f e d c b a
1 1 1 1 1 0 1
↓
7D



g f e d c b a
0 0 0 0 1 1 1
↓
7

g f e d c b a
1 0 1 1 0 1 1
↓
5B

g f e d c b a
1 1 0 0 1 1 0
↓
66

This page is intentionally left blank for writing code

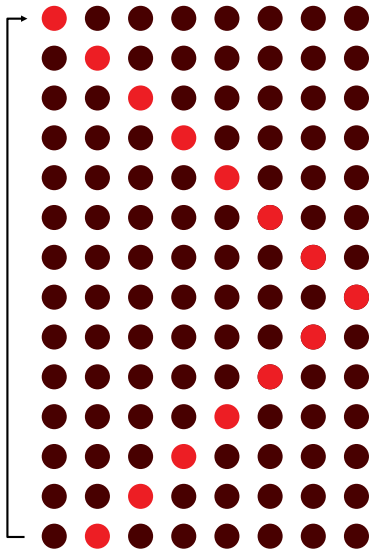
```
1   .ORG 0x00             ; Set the origin to address 0x00 (start of flash memory)
2
3       LDI R16, 0xFF      ; Load immediate value 0xFF into register R16
4       OUT DDRD, R16      ; Set all pins of Port D as outputs (DDRD = 0xFF)
5
6   AGAIN1:                ; Label for the main loop
7       LDI ZH, HIGH(0x400) ; Load the high byte of the address of MYDATA into ZH
8       LDI ZL, LOW(0x400)  ; Load the low byte of the address of MYDATA into ZL
9       LDI R17, 10         ; Load loop counter (10 iterations) into R17
10
11  AGAIN2:                ; Label for the inner loop
12      LPM R16, Z+        ; Load program memory byte from the address in Z (ZL:ZH) into R16, then increment Z
13      OUT PORTD, R16      ; Output the value in R16 to Port D (controlling LEDs or a display)
14      CALL DELAY          ; Call the delay subroutine
15      DEC R17            ; Decrement the loop counter R17
16      BRNE AGAIN2        ; Branch if R17 is not zero (loop back to AGAIN2)
17      JMP AGAIN1         ; Jump back to the beginning of the main loop (AGAIN1)
18
19  ; 2-second delay subroutine
20  DELAY:
21      LDI R23, 64
22  L1:
23      LDI R24, 200
24  L2:
25      LDI R25, 250
26  L3:
27      NOP
28      NOP
29      DEC R25
30      BRNE L3
31      DEC R24
32      BRNE L2
33      DEC R23
34      BRNE L1
35      RET
36
37  .ORG 0x200
38
39  MYDATA:                ; My student ID hereeeee...
40      .DB 0x7D, 0x7D, 0x5B, 0x5B, 0x07, 0x07, 0x5B, 0x66, 0x5B, 0x5B
```

$$\left[\,1+\left(\left\{1+\left[\left\{1+(5\times250)-1+3\right\}\times200\right]-1+3\right\}\times X\right)-1+4\,\right]\times\frac{1}{8M}=2\ \text{seconds}$$

$$X \approx 64$$

Problem 7

Write a complete assembly program using macros to create a LED sequence on 8 LEDs connected to PORTD (PD0-PD7). The LEDs should light up in a pattern where a single LED appears to move back and forth across the row of LEDs. Each position change should have approximately 0.5 seconds delay. (assuming the microcontroller is running at 16MHz)



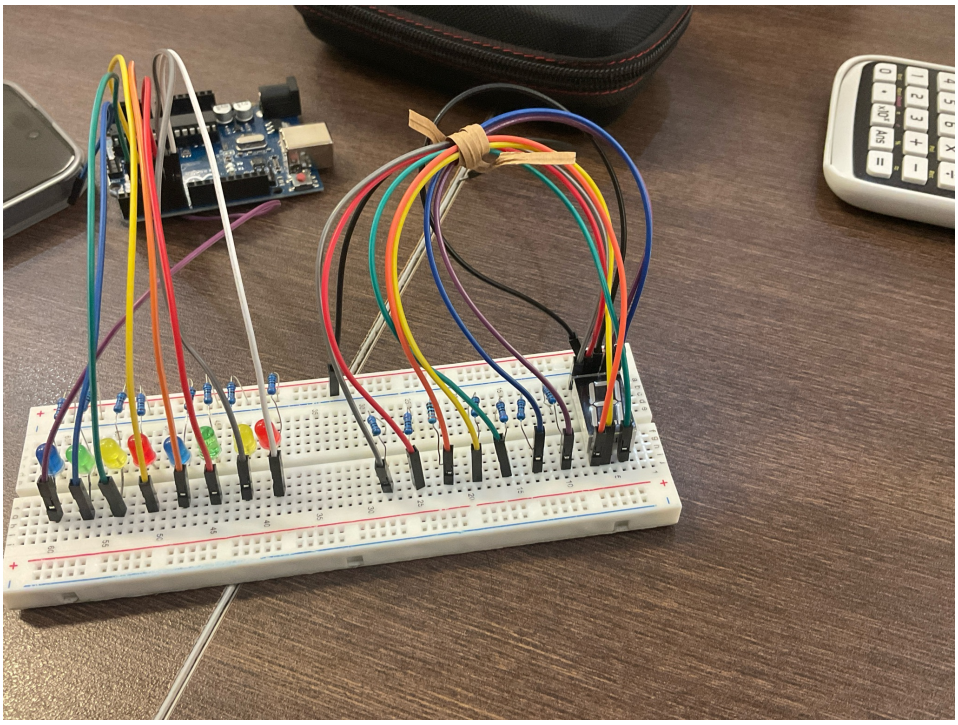
Use the following macros in your program:

```
.MACRO DELAY
    LDI R20, @0
L1: LDI R21, @1
L2: LDI R22, @2
L3: NOP
    NOP
    DEC R22
    BRNE L3
    DEC R21
    BRNE L2
    DEC R20
    BRNE L1
.ENDMACRO
```

Draw a complete circuit diagram showing the connection of 8 LEDs to PORTD (PD0-PD7) of the Arduino Uno board as described in the problem. Your diagram should include:

1. Arduino Uno board
2. Eight LEDs connected to the pins corresponding to PD0 through PD7
3. Current-limiting resistors for each LED (specify resistance value)
4. Power connections if needed

Note: Arrange the LEDs in a horizontal row to match the sequential pattern implemented in your assembly program.



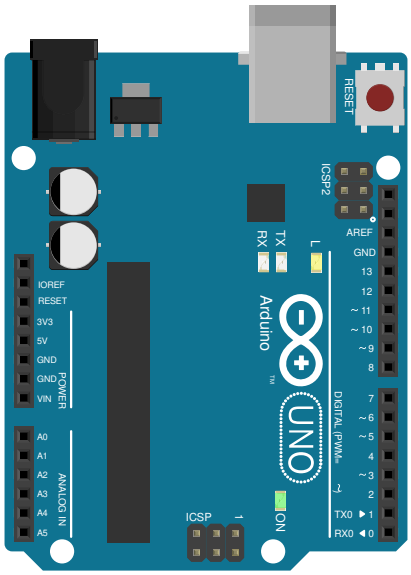
Hint: *Use LSL/LSR to shift a single '1' bit left/right to move the LED position, and implement a direction flag (0 or 1) that changes when the LED reaches either PD0 or PD7 to create the back-and-forth effect. Use DELAY X, 200, 250, Calculate X to get approximately 0.5 seconds delay*

```
.ORG 0x00

.MACRO DELAY
    LDI  R20, @0
L1: LDI  R21, @1
L2: LDI  R22, @2
L3: NOP
    NOP
    DEC  R22
    BRNE L3
    DEC  R21
    BRNE L2
    DEC  R20
    BRNE L1
.ENDMACRO

.MACRO SETLED
    LDI  R16, @0
    OUT  PORTD, R16
.ENDMACRO

MAIN:
    ; Initialize stack pointer
    LDI  R16, HIGH(RAMEND)
    OUT  SPH, R16
    LDI  R16, LOW(RAMEND)
    OUT  SPL, R16

    ; Initialize PORTD as output
    LDI  R16, 0xFF       ; All pins as output
    OUT  DDRD, R16

    ; Initialize direction flag and LED pattern
    LDI  R18, 0x01       ; Direction: 1=right, 0=left
    LDI  R17, 0x01       ; Start with LED at PD0

LOOP:
    ; Output current LED pattern
    MOV  R16, R17
    OUT  PORTD, R16

    ; Delay to make movement visible
    DELAY 32, 200, 250

    ; Check direction
    CPI  R18, 0x01
    BRNE GO_LEFT

GO_RIGHT:
    ; Shift LED to the right
    LSL  R17

    ; Check if we've reached PD7
    CPI  R17, 0x80
    BRNE LOOP            ; If not at PD7, continue in same directi

    ; At PD7, change direction
    LDI  R18, 0x00       ; Set direction to left
    RJMP LOOP

GO_LEFT:
    ; Shift LED to the left
    LSR  R17

    ; Check if we've reached PD0
    CPI  R17, 0x01
    BRNE LOOP            ; If not at PD0, continue in same directi

    ; At PD0, change direction
    LDI  R18, 0x01       ; Set direction to right
    RJMP LOOP
```

Appendix

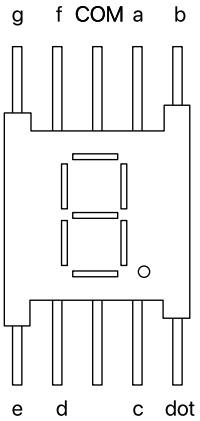
Arduino UNO Pin Layout

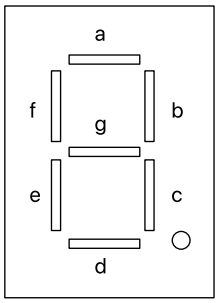
Label	Port
A0	PC0(ADC0)
A1	PC1(ADC1)
A2	PC2(ADC2)
A3	PC3(ADC3)
A4	PC4(ADC4)
A5	PC5(ADC5)



Label	Port
SCL	PC5(ADC5/SCL)
SDA	PC4(ADC4/SDA)
AREF	AREF
GND	GND
13	PB5(SCK)
12	PB4(MISO)
11	PB3(MOSI)
10	PB2(OC1B)
9	PB1(OC1A)
8	PB0
7	PD7
6	PD6
5	PD5
4	PD4
3	PD3(INT1)
2	PD2(INT0)
1	PD1(TXD)
0	PD0(RXD)

7-Segment Display Pinout





Number	g	f	e	d	c	b	a
0	0	1	1	1	1	1	1
1	0	0	0	0	1	1	0
2	1	0	1	1	0	1	1
3	1	0	0	1	1	1	1
4	1	1	0	0	1	1	0
5	1	1	0	1	1	0	1
6	1	1	1	1	1	0	1
7	0	0	0	0	0	1	1
8	1	1	1	1	1	1	1
9	1	1	0	1	1	1	1

Mnemonics	Operands	Description	Operation	Flags	#Clock
Arithmetic and Logic Instructions					
ADD	Rd, Rr	Add two registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with carry	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl, K	Add immediate to word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract immediate	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with carry	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with carry imm.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl, K	Subtract imm. from word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND	Rd ← Rd × Rr	Z,N,V	1
ANDI	Rd, K	AND with immediate	Rd ← Rd × K	Z,N,V	1
OR	Rd, Rr	Logical OR	Rd ← Rd ∨ Rr	Z,N,V	1
ORI	Rd, K	OR with immediate	Rd ← Rd ∨ K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's complement	Rd ← 0xFF - Rd	Z,C,N,V	1
NEG	Rd	Two's complement	Rd ← 0x00 - Rd	Z,C,N,V,H	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V	1
Branch Instructions					
RJMP	k	Relative jump	PC ← PC + k + 1	None	2
JMP	k	Direct jump	PC ← k	None	3
RCALL	k	Relative call	PC ← PC + k + 1	None	3
CALL	k	Direct call	PC ← k	None	4
RET		Return from subroutine	PC ← STACK	None	4