Microprocessor Principles and Applications

Midterm

Name:_____ Fall 2023

ID:_____

The exam is 120 minutes long. The total score is 102pts. Please read questions carefully.

1. Terminologies (32pts)

   (1). File select register

   FSRs are basically memory pointers. They are typically used for handling arrays and pointer-based data accessing in the data memory.

   (2). Access bank

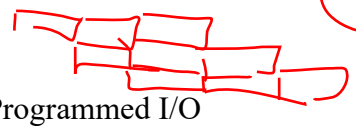
   To facilitate access for the most commonly used data memory locations, the data memory is configured with an "Access Bank", which allows users to access a mapped block of memory without bank switching.

   (3). Harvard architecture (Hint: One type of CPU architecture)

   It is a type of CPU architecture which uses separate program and data memory units along with separate buses for program and data. This means that these processors can execute programs and access data simultaneously.

   (4). Pipeline

   Pipelining is a technique that overlaps instruction fetch (instruction read) with execution.

   (5). Programmed I/O

   A microcontroller communicates with an external device via one or more registers called I/O ports using programmed I/O. Each bit in the port can be configured individually as either input or output.

   (6). Program counter (PC)

   The program counter normally contains the address of the next instruction to be executed.

   (7). Microcontroller

   Typical microcontrollers include a CPU, memory, I/O, along with certain peripheral functions such as timers, and ADC (Analog-to-Digital Converter) all in a single chip.

   (8). Binary-Coded-Decimal digits (BCD)

   The 10 decimal digits 0 through 9 can be represented by their corresponding 4-bit binary numbers. The digits coded in this fashion are called binary-coded-decimal digits in 8421 code, or BCD digits.

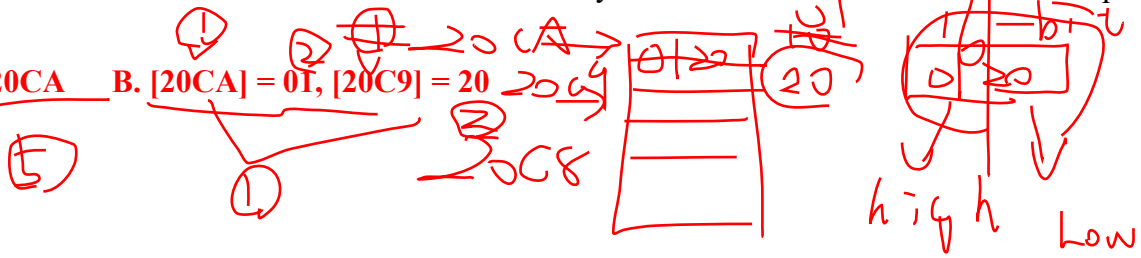
2. What is the difference between memory mapping in a microcontroller and memory-mapped I/O? (10pts)

Ans:

Memory-mapping provides the physical addresses for the microcomputer's main memory while memory-mapped I/O maps port addresses into memory locations.

3. Suppose that an 8-bit microcontroller has a 16-bit stack pointer and uses a 16-bit register to access the stack from the **BOTTOM**. Assume that initially the stack pointer and the 16-bit register contain $20C8_{16}$ and 0120_{16} respectively. After the PUSH operation: (10pts)
   A. What are the contents of the stack pointer?
   B. Suppose the basic unit of data memory is 8 bits, what are the memory locations to store the pushed data value? What are the contents of these memory locations if little-endian format is adopted?

Ans: A. 20CA B. [20CA] = 01, [20C9] = 20



4. Given a PIC18F assembly language program like the following code. Please answer the following questions. (10pts)
   A. What is the address where the translated machine code of "CLRF SIGN1" is placed in the program memory?
   B. Determine the contents of address 0x204 after assembling the following code.

```
        INCLUDE <PICF4321.INC>
        ORG 0x100
MULT1   EQU 0x15
MULT2   EQU 0x17
SIGN1   EQU 0x50
SIGN2   EQU 0x51
        MOVLW 0xFE
        MOVWF MULT1
        MOVLW 0xFC
        MOVWF MULT2
        CLRF SIGN1

        ORG 0x200
        DW 2023H, 1026H, 1157H, DE22H
```

Ans: (1) 0x108

(2) 57H

5. Write a PIC18F assembly language program at address 0x100 to move a block of 8-bit data of length 10 from the source block (from HIGH to LOW address) starting at data register address 0x55 to the destination block (from LOW to HIGH addresses) starting at data register address 0x30. That is, [0x55] will be moved to [0x30], [0x54] to [0x31], and so on. Assume that data for the source block and the destination block are already stored in data memory addresses. (10pts)

```
            INCLUDE <P18F4321.INC>
COUNTER     EQU     0x80
            ORG     0x100
            MOVLW   D'10'       ; Move 10 decimal into WREG
            MOVWF   COUNTER     ; Initialize Counter reg (0x80) with 10 decimal
            LFSR    0, 0x0055   ; Initialize FSR0 with source starting address
            LFSR    1, 0x0030   ; Initialize FSR1 with destination starting address
BACK        MOVFF   POSTDEC0, POSTINC1  ; Move Source data to destination
            DECF    COUNTER, F  ; Decrement Counter by 1
            BNZ     BACK        ; Branch to  BACK if  Zero flag = 0, otherwise
                                ; go to the next instruction
            SLEEP               ; HALT
            END
```

6. Write a PIC18F assembly language program at address 0x100 to add two 8-bit numbers (N1 and N2). Data register 0x20 contains N1. The low four bits of N2 are stored in the upper nibble of data register 0x21 while the high four bits of N2 are stored in the lower nibble of data register 0x21. Store result in data register 0x30. (10pts)

Ans:

```
Assume N1 and N2 are already loaded into registers 0x20 and 0x21 respectively.
INCLUDE <PIC18F4520.INC>
ORG 0x100
SWAPF 0x21, F ; Swap nibbles of N2 in 0x21
MOVF  0x20, W ; Move [0x20] into WREG
ADDWF 0x21, W ; Add [WREG] with [0x21], store result in WREG
MOVWF 0x30 ; Store result in 0x30
SLEEP
END
```

7. Write a PIC18F assembly language program at address 0x100 to multiply an 8-bit unsigned number in data register 0x50 by 16. Store the 8-bit result in WREG. Do not use any multiplication instructions. Use ROTATE instruction. Assume that a '1' is not shifted out of the most significant bit each time after rotating to the left. Also, assume that the 8-bit unsigned number is already loaded into data register 0x40. Write the program using a loop. (10pts)

```
            INCLUDE  <P18F4321.INC>
            ORG      0x100
COUNTER     EQU      0x70
            MOVLW    4              ; Initialize COUNTER with 4
BACK        BCF      STATUS, C      ; Clear Carry
            RLCF     0x50, F        ; Rotate [0x50] four times
            DECF     COUNTER, F     ; to left to multiply [0x50] by 16
            BNZ      BACK           ; branch to BACK if Z = 0
            MOVF     0x50, W        ; Move result to WREG
FOREVER     GOTO     FOREVER        ; Stop
            END
```

8. Write a PIC18F assembly language program at address 0x100 to add two 16-bit numbers as follows.

```
            [0x51] [0x50]
PLUS        [0x61] [0x60]
-------------------------------------
            [0x61] [0x60]
```

Assume data registers 0x50 and 0x51 contain low and high bytes of the first 16-bit numbers while data registers 0x60 and 0x61 contain the second 16-bit numbers. Also, assume that data are already loaded into the data registers. (10pts)

Ans:

```
INCLUDE   <P18F4321.INC>
ORG       0x100
MOVF      0x50, W     ; Move low byte of first data into WREG
ADDWF     0x60, F     ; Add low 8 bits, store result in 0x60
MOVF      0x51, W     ; Move high byte of first data into WREG
ADDWFC    0x61, F     ; Add high bytes with carry, store result in 0x61
SLEEP                 ; Halt
END
```

BTFSC F, b, a	BTFSC 0x50, 3	If bit number 3 in data register 0x50 is 0, skip the next instruction; otherwise, the next instruction is executed.
BTFSS F, b, a	BTFSC 0x40, 0	If bit number 0 in data register 0x40 is 1, skip the next instruction; otherwise, the next instruction is executed.
BTG F, b, a	BTG 0x20, 2	Invert (ones complement) bit number 2 in data register 0x20.
BZ d8	BZ START	Branch to START if Z = 1 where START is an 8-bit signed #
CALL k, s	CALL BEGIN	This is a two-word instruction. The simplest way to CALL a subroutine is when s = 0 (default); pushes current program counter (PC+4) which is also the return address , and loads PC with BEGIN which is the starting address of the subroutine.
CLRF F, a	CLRF 0x40	Clear the contents of data register to 0.
CLRWDT	CLRWDT	Reset the watchdog timer.
COMF F, d, a	COMF 0x30, F	One's complement each bit of [0x30], and store the result in 0x30.
	COMF 0x30, W	One's complement each bit of [0x30], and store the result in WREG.
CPFSEQ F, a	CPFSEQ 0x30	Unsigned comparison. If [0x30] =[WREG], skip the next instruction; else, the next instruction is executed.
CPFSGT F, a	CPFSGT 0x50	Unsigned comparison. If [0x50] >[WREG], skip the next instruction; else, the next instruction is executed.
CPFSLT F, a	CPFSLT 0x60	Unsigned comparison. If [0x60] <[WREG], skip the next instruction; else, the next instruction is executed.
DAW	DAW	Decimal Adjust [WREG] resulting from earlier addition of two packed BCD digits providing correct packed BCD result.
DECF F, d, a	DECF 0x20, W	Decrement [0x20] by 1, and store result in WREG.
	DECF 0x20, F	Decrement [0x20] by 1, and store result in 0x20.

Instruction	*Example*	*Operation*
ADDLW data8	ADDLW 0x07	[WREG] + 0x07 t [WREG]
ADDWF F, d, a	ADDWF 0x20, W	[WREG] + [0x20] → [WREG]
	ADDWF 0x20, F	[WREG] + [0x20] → [0x20]
ADDWFC F, d, a	ADDWFC 0x40, W	[WREG] + [0x40] + Carry → [WREG]
	ADDWFC 0x40, F	[WREG] + [0x40] + Carry → [0x40]
ANDLW data8	ANDLW 0x02	[WREG] AND 0x02 → [WREG]
ANDWF F, d, a	ANDWF 0x30, W	[WREG] AND [0x30] → [WREG]
	ANDWF 0x30, F	[WREG] AND [0x30] → [0x30]
BC d8	BC START	Branch to START if C = 1 where START is an 8-bit signed #
BCF F, b, a	BCF 0x30, 2	Clear bit number 2 to 0 in data register 0x30, store result in 0x30
	BCF STATUS, C	Clear the Carry Flag to 0 in the status register.
BN d8	BN START	Branch to START if N = 1 where START is an 8-bit signed #
BNC d8	BNC START	Branch to START if C = 0 where START is an 8-bit signed #
BNN d8	BNN START	Branch to START if N = 0 where START is an 8-bit signed #
BNOV d8	BBNOV START	Branch to START if OV = 0 where START is an 8-bit signed #
BNZ d8	BNZ START	Branch to START if Z = 0 where START is an 8-bit signed #
BOV d8	BOV START	Branch to START if OV = 1 where START is an 8-bit signed #
BRA d8	BRA START	Branch always to START where START is an 8-bit signed #
BSF F, b, a	BSF 0x20, 7	Set bit number 7 to 1 in data register 0x20, store result in 0x20.
	BSF STATUS, C	Set the Carry Flag to 1 in the status register.

IORWF F, d, a	IORWF 0x50, W	The contents of WREG are logically ORed with the contents of 0x50, and the result is stored in WREG.
	IORWF 0x50, F	The contents of WREG are logically ORed with the contents of 0x50, and the result is stored in 0x50.
LFSR F, k	LFSR 0, 0x0080	Load 00H into FSR0H, and 80H into FSR0L.
MOVF F, d, a	MOVF 0x30, W	The contents of 0x30 are loaded into WREG.
	MOVF 0x30, F	The contents of 0x30 are copied into 0x30.
MOVFF Fs, Fd	MOVFF 0x50,0x60	Move [0x50] into [0x60]. The contents of 0x50 are unchanged.
MOVLB k	MOVLF 0x04	Load BSR with 04H.
MOVLW k	MOVLW 0x21	Load WREG with 21H.
MOVWF F, a	MOVWF 0x50	Move the contents of WREG into 0x50.
MULLW k	MULLW 0xF1	[WREG] x F1H → [PRODH] [PRODL]; unsigned multiplication.
MULWF F, a	MULWF 0x30	[WREG] x [0x30] → [PRODH] [PRODL] ; unsigned multiplication.
NEGF F, a	NEGF 0x20	Negate the contents of 0x20 using two's complement.
NOP	NOP	No Operation.
POP	POP	Discard top of stack pointed by SP, and decrement PC by 1.
PUSH	PUSH	Push or write PC onto the stack, and increment SP by 1.
RCALL n	RCALL START	Relative subroutine CALL. One-word instruction. Pushes PC+2 onto the hardware stack. START is an 11-bit signed number. Jumps to a subroutine located at an address (PC +2) + 2 x START.
RESET	RESET	Reset all registers and flags that are affected by a MCLR reset.
RETFIE	RETFIE	Return from Interrupt.
RETLW k	RETLW k	WREG is loaded with the 8-bit literal k, and PC is loaded with the return address from the top of stack.
RETURN	RETURN	Return from subroutine.

Instruction	*Example*	*Operation*
DECFSZ F, d, a	DECFSZ 0x30, W	Decrement [0x30] by 1, and store result in WREG. If [WREG] = 0, skip the next instruction; else, execute the next instruction.
	DECFSZ 0x30, F	Decrement [0x30] by 1, and store the result in 0x30. If [0x30] = 0, skip the next instruction; else, execute the next instruction.
DECFSNZ F, d, a	DECFSNZ 0x50,W	Decrement [0x50] by 1, and store result in WREG. If [WREG] ≠ 0, skip the next instruction; else, execute the next instruction.
	DECFSNZ 0x50, F	Decrement [0x50] by 1, and store the result in 0x50. If [0x50] ≠ 0, skip the next instruction; else, execute the next instruction.
GOTO k	GOTO START	Unconditional branch to address START.
INCF F, d, a	INCF 0x20, W	Increment [0x20] by 1, and store result in WREG.
	INCF 0x20, F	Increment [0x20] by 1, and store result in 0x20.
INCFSZ F, d, a	INCFSZ 0x30, W	Increment [0x30] by 1, and store result in WREG. If [WREG] = 0, skip the next instruction; else, execute the next instruction.
	INCFSZ 0x30, F	Increment [0x30] by 1, and store the result in 0x30. If [0x30] = 0, skip the next instruction; else, execute the next instruction.
INCFSNZ F, d, a	INCFSNZ 0x50, W	Increment [0x50] by 1, and store result in WREG. If [WREG] ≠ 0, skip the next instruction; else, execute the next instruction.
	INCFSNZ 0x50, F	Increment [0x50] by 1, and store the result in 0x50. If [0x50] ≠ 0, skip the next instruction; else, execute the next instruction.
IORLW k	IORLW 0x54	The contents of WREG are logically ORed with 0x54, and the result is stored in WREG.

Instruction	Example	Operation
RLCF F, d, a	RLCF 0x20, W	Rotate [0x20] once to the left through Carry. Store result in WREG.
	RLCF 0x20, F	Rotate [0x20] once to the left through Carry. Store result in register 0x20.
RLNCF F, d, a	RLNCF 0x30, W	Rotate [0x30] once to the left without Carry. Store result in WREG.
	RLNCF 0x30, F	Rotate [0x30] once to the left without Carry. Store result in register 0x30.
RRCF F, d, a	RRCF 0x50, W	Rotate [0x50] once to the right through Carry. Store result in WREG.
	RRCF 0x50, F	Rotate [0x50] once to the right through Carry. Store result in register 0x50.
RRNCF F, d, a	RRNCF 0x60, W	Rotate [0x60] once to the right without Carry. Store result in WREG.
	RRNCF 0x60, F	Rotate [0x60] once to the right without Carry. Store result in register 0x60.
SETF F, a	SETF 0x30	The contents of 0x30 are set to 1's.
SLEEP	SLEEP	Enter Sleep mode.
SUBFWB F, d, a	SUBFWB 0x20, W	[WREG] –[0x20] – Carry → [WREG]
	SUBFWB 0x20, F	[WREG] –[0x20] – Carry → [0x20]
SUBLW k	SUBLW 0x05	[0x05] – [WREG] → [WREG]
SUBWF F, d, a	SUBWF 0x50, W	[0X50] – [WREG] → [WREG]
	SUBWF 0x50, F	[0X50] – [WREG] → [0x50]
SUBWFB F, d, a	SUBWFB 0x32, W	[0x32] – [WREG] – Carry → [WREG]
	SUBWFB 0x32, F	[0x32] – [WREG] – Carry → [0x32]
SWAPF F, d, a	SWAPF 0x30, W	The upper and lower 4 bits of register 0x30 are exchanged. The result is stored in WREG.
	SWAPF 0x30, F	The upper and lower 4 bits of register 0x30 are exchanged. The result is stored in register 0x30.
TBLRD	TBLRD	Table Read.
TBLWT	TBLWT	Table Write.
TSTFSZ F, a	TSTFSZ 0x50	If [0x50] = 0, skip the next instruction; else, execute the next instruction.
XORLW k	XORLW 0xF2	[WREG] XOR F2H → [WREG]
XORWF F, d, a	XORWF 0x30, W	[WREG] XOR [0x30] → [WREG]
	XORWF 0x30, F	[WREG] XOR [0x30] → [0x30]

第一題

1. 有表達出是一個 pointer 或是指標 +2
   紀錄 Data memory address 作操作+2
   如果只寫 address 沒有說是哪裡的 address -1

2. 減少 bank switch 次數（或說是預設的 bank ，可以提高效率等等的）+3 其他說明+1
   把 access 解讀成動詞來解釋 access 一個 bank 0 分
   或是只解釋 bank 也不會拿到分數

3. Program memory 和 data memory 分開的架構+3
   可同時讀取兩者的資料或同時操作+1（或是寫出有分別的 bus)
   畫圖的分數就照上面的給分，有畫出來就拿到分

4. overlap (+3) instructions (+1)

5. one or more registers called I/O ports (+3)
   each bit can be configured input or output (+1)
   其他回答根據投影片部份給分

6. address (+1) of the next instruction (+3)

7. 僅提到 computer +1
   有列舉至少 2 個元件（CPU, memory, I/O, along with certain peripheral functions such as timers, and ADC）+2
   有提到 single chip +1

8. 有提到「是以二（十六）進位表示十進位」+2
   有提到用來表示 0~9 或是十進位的各個位數+1
   有提到 BCD 的額外功能（如處理 I/O、需要注意進位、處理效率慢）+1

第二題

memory mapping 有寫到提供 physical address +5
memory mapped 有寫到將 Port address 放到 memory +5

第三題
A. 答案對 +5
B. 位置寫對 +1 順序寫對間隔為 1 +4

第四題
兩小題都是答案對 +5

第五題
沒 include org sleep/end -1
語法錯-1
位置放錯 -1
邏輯/結果錯-4

第六題
沒 include -1
沒 org -1
沒 sleep 或 end -1
邏輯錯誤(沒 swap) -4
正確性 -2
語法錯誤 -1

第七題
沒有 Include -1
沒有 end -1
答案沒存回 wreg -3
忘記清掉 Carry -1
沒使用到 loop -10
邏輯錯誤 -3

第八題
沒有 Include -1
沒有 org -1
沒有 end -1
邏輯大致正確，差一句扣正確性 -2
比較多有問題 扣邏輯-4 和正確性-2，總共-6