Microprocessor Principles and Applications –Midterm
Nov. 4, 2020

1. (a) (5%) What does pipelining mean?
   (b) (10%) Assuming signed numbers, find the sign, carry, zero, and overflow flags of:
   (1) $09_{16} + 17_{16}$ (2) $6E_{16} + 3A_{16}$

2. (10%) What is the basic difference between standard I/O and memory-mapped I/O? Identify the programmed I/O technique used by the PIC18F.

3. (10%) Assume 2 two's-complement signed numbers, $M=1100_2$ and $Q=0111_2$. Perform signed multiplication. Please write down the details of how you perform the signed multiplication, and clearly show the result of product.

4. (a) (5%) What is the basic difference between: Program Counter (PC) and *Function Select Registers* (FSRs)?
   (b) (5%) What is the advantage of incorporating the "Access Bank" in the *PIC18F*?

5. (15%) 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.

6. (15%) Write a PIC18F assembly language at address 0x100 program to divide an unsigned 16-bit number by 2. Assume that the higher byte of the 16-bit number is already stored in data register [0x20], and lower byte in data register [0x21]. Discard remainder.

7. (15%) Write a PIC18F assembly language program at address 0x100 to add a 4-bit unsigned number stored in the high nibble of data register 0x30 with a 4-bit unsigned number stored in the low nibble of data register 0x30. Store the 8-bit result in 0x30.

8. (10%) Calculate the instruction cycles executed in this macro. Please present the result in terms of X and Y. (Y is a number less than 80)

```
DELAY macro X, Y
      local LOOP1   ;prevent compile error
      local LOOP2
      movlw Y
      movwf REG2
rlncf REG2
LOOP2:
      movlw X
      movwf REG1
LOOP1:
      decfsz REG1,1
            goto LOOP1
      decfsz REG2,1
            goto LOOP2
      endm
```

2Y

2 + 2Y (2 + 3X + 3)

Instruction	Number of cycle to execute
movlw	1
movwf	1
rlncf	1
decfsz	1, 2(If skip a 1-word instruction ex: bra) 3(If skip a 2-word instruction ex: goto),
goto	2

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.

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
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.

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	MOVLB 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
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 FF.
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 0x 30.
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]