Byeongchan Gwak, Student ID : 501026

Washington University in St.Louis

McKelvey School of Engineering

Fall Semester 2021

CSE467M: Embedded Computing Systems

Midterm Exam #1

1. **What is the definition of instruction set? (10 points)**

   An instruction set is a group of commands for a CPU in machine language. It is the programmer's interface to the hardware.

2. **A project that you work on as assigned designer has requirement for large amount of data streaming complex arithmetic function application. Which type of the CPU would be the best selection for the project:**

   (a) Microcontroller (PIC16F for example)
   (b) Microprocessor (ARM Cortex M-7 for example)
   (c) Digital Signal Processor (C55x or C64x for example).

   Explain your reasoning. (10 points)

   The answer is (c)Digital Signal Processor. The C55x is an accumulator architecture which means that one of the operands is the accumulator, so it need not be specified in the instruction. Accumulator-oriented instructions are also well-suited to the types of operations performed in digital signal processing, such as a1x1 + a2x2 + … And the C64x is VLIW(very long instruction word) DSP which is proper for data streaming.

3. **Which computer architecture, Neumann Architecture or Harvard Architecture is used for Digital Signal Processors and why is that the case? (10 points)**

   The answer is Harvard Architecture because of the separation of program and data memories which provides higher performance for digital signal processing. Having two memories with separate ports provides higher memory bandwidth as well as not making data and memory compete for the same port.

4. **ARM processor's Instruction Set does not have a division arithmetic instruction. Designers need to figure out some other way to perform division routine in ARM Assembly code.**
   (a) Write an ARM assembly language routine that would divide two integer numbers 240 and 4. Comment all your steps of the code.
   (b) Write an ARM assembly language routine that would divide two integer numbers 240 and 6. Comment all your steps of the code.
   (c) Write an ARM assembly language routine that would divide any two integer numbers. If there is the division reminder store it in a register at the end. Please take into account potential divide-by-zero exceptions. Comment all your steps of the code. (30 points)

I'm going to implement division with subtraction. How many times can a dividend be subtracted by a divisor?

(a) Write an ARM assembly language routine that would divide two integer numbers 240 and 4. Comment all your steps of the code.

```
.INIT:
        mov   r1, #0            ; set r1 to number 0
        mov   r2, #240          ; set r2 for constant number 240
        mov   r3, #4            ; set r3 for constant number 4

.TEST:
        cmp   r2, r3            ; compare r2 with r3
        blt   LOOPEND           ; if r2 < r3, exit loop

.BODY:
        sub   r2, r2, r3        ; r2 = r2 - r3
        add   r1, r1, #1        ; r1 = r1 + 1
        b     TEST              ; go to TEST

.LOOPEND:
        str   r10, r1           ; store quotient into register r10
```

(b) Write an ARM assembly language routine that would divide two integer numbers 240 and 6. Comment all your steps of the code.

```
.INIT:
        mov   r1, #0            ; set r1 to number 0(quotient)
        mov   r2, #240          ; set r2 for constant number 240
        mov   r3, #6            ; set r3 for constant number 6

.TEST:
        cmp   r2, r3            ; compare r2 with r3
        blt   LOOPEND           ; if r2 < r3, exit loop

.BODY:
        sub   r2, r2, r3        ; r2 = r2 - r3
        add   r1, r1, #1        ; r1 = r1 + 1
        b     TEST              ; go to TEST

.LOOPEND:
        str   r10, r1           ; store quotient into register r10
        str   r11, r2           ; store remainder into register r11
```

(c) Write an ARM assembly language routine that would divide any two integer numbers. If there is the division reminder store it in a register at the end. Please take into account potential divide-by-zero exceptions. Comment all your steps of the code. (30 points)

- **Let's say two integer a and b.**

```
.INIT:
        mov   r1, #0          ; set r1 to number 0(quotient)
        mov   r2, #0          ; set r2 to number 0(remainder)
        adr   r4, a           ; load r4 with address of a(dividend)
        adr   r5, b           ; load r5 with address of b(divisor)

        ldr   r3, [r5, #0]    ; get value of b(divisor)
        cmp   r3, #0          ; compare divisor with number 0
        beq   LOOPEND         ; if divisor == 0, exit loop

        ldr   r2, [r4, #0]    ; get value of a(dividend)

.TEST:
        cmp   r2, r3          ; compare r2 with number r3
        blt   LOOPEND         ; if r2 < r3, exit loop

.BODY:
        sub   r2, r2, r3      ; r2 = r2 - r3
        add   r1, r1, #1      ; r1 = r1 + 1
        b     TEST            ; go to TEST

.LOOPEND:
        str   r10, r1         ; store quotient into register r10
        str   r11, r2         ; store remainder into register r11
```

5. **Why do most programs use interrupt-driven I/O over busy/wait? (10 points)**

   Busy-wait I/O is inefficient because CPU does nothing but test the device status while the I/O transaction is in progress. However, the CPU could do useful work in parallel with the interrupt-drive I/O.

6. **Which cache type is more suited for Embedded System design**

   (a) Direct-mapped Cache
   (b) Set associate Cache

   Explain your reasoning. (10 points)

   The answer is (b) Set associate Cache. In Embedded System design, execution speed is very important for the performance of the system. Memories are comparatively very slow to CPU so caches are widely used to speed up reads and writes in memory systems. Due to the nature of the Direct-mapped Cache, it does have limits in its caching power. Whereas the Set associate Cache generally provides higher hit rates than the Direct-mapped Cache because conflicts between a small set of locations can be resolved within the cache.

7. **If we want an average memory access time of 8 ns, our cache memory access is 4 ns, and our main memory access time is 70 ns, what cache hit rate must we achieve? (10 points)**

   a. **Hit ratio : A**
   b. **A * 4ns + (1-A)*70ns = 8ns**
   c. **66A = 62**
   d. **A = 0.939393…**
   e. **Almost 94% cache hit rate should be achieved**

Byeongchan Gwak, Student ID : 501026

8. Cache memory analysis (30 points)

Memory contents using a 3-bit memory address scheme is shown below:

Address	Data
000	1111
001	0000
010	1000
011	0001
100	1100
101	0011
110	1010
111	0101

Apply the following pattern of addresses 111, 000, 101, 010, 110, 001, 100, 011 and show step-by-step the state of the cache after each memory access for:

(a) Direct-mapped cache with 4 blocks

1.After 111 access:

Block	Tag	Data
00	-	-
01	-	-
10	-	-
11	1	0101

2.After 000 access:

Block	Tag	Data
00	0	1111
01	-	-
10	-	-
11	1	0101

3.After 101 access:

Block	Tag	Data
00	0	1111
01	1	0011
10	-	-
11	1	0101

4.After 010 access:

Block	Tag	Data
00	0	1111
01	1	0011
10	0	1000
11	1	0101

5.After 110 access:

Block	Tag	Data
00	0	1111
01	1	0011
10	1	1010
11	1	0101

6.After 001 access:

Block	Tag	Data
00	0	1111
01	0	0000
10	1	1010
11	1	0101

7.After 100 access:

Block	Tag	Data
00	1	1100
01	0	0000
10	1	1010
11	1	0101

8.After 011 access:

Block	Tag	Data
00	1	1100
01	0	0000
10	1	1010
11	0	0001

Byeongchan Gwak, Student ID : 501026

(b) Two-way (banks) with 4 blocks each set associative cache with least-recently replacement (LRU) policy

1.After 111 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	-			-
01	-			-
10	-			-
11	1	0101		-

(a)

2.After 000 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	1111		-
01	-			-
10	-			-
11	1	0101		-

3.After 101 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	1111		-
01	1	0011		-
10	-			-
11	1	0101		-

4.After 010 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	1111		-
01	1	0011		-
10	0	1000		-
11	1	0101		-

5.After 110 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	1111		-
01	1	0011		-
10	0	1000	1	1010
11	1	0101		-

6.After 001 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	1111		-
01	1	0011	0	0000
10	0	1000	1	1010
11	1	0101		-

7.After 100 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	1111	1	1100
01	1	0011	0	0000
10	0	1000	1	1010
11	1	0101		-

8.After 011 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	1111	1	1100
01	1	0011	0	0000
10	0	1000	1	1010
11	1	0101	0	0001

Byeongchan Gwak, Student ID : 501026

(c) Two-way (banks) with 2 blocks each set associative with least-recently replacement (LRU) policy.

1.After 111 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	-	-	-	-
1	11	0101	-	-

2.After 000 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	00	1111	-	-
1	11	0101	-	-

3.After 101 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	00	1111	-	-
1	11	0101	10	0011

4.After 010 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	00	1111	01	1000
1	11	0101	10	0011

5.After 110 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	11	1010	01	1000
1	11	0101	10	0011

6.After 001 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	11	1010	01	1000
1	00	0000	10	0011

7.After 100 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	11	1010	10	1100
1	00	0000	10	0011

8.After 011 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	11	1010	10	1100
1	00	0000	01	0001

9. C code for for-loop running on an ARM processor is defined as:

for (i = 0, f = 0, g = 0; i<10; i++)

{

f = f + a[i]*b[i] + c[i]*d[i] – e[i];

g = g + (a[i]+d[i])*(b[i] – c[i])*e[i];

}

Using Example 3.10 from the textbook as a staring reference to write the for-loop shown above in ARM assembly language and determine total execution time for the loop in the cycles. (40 points)

```
.INIT:
        mov   r0, #0          ; set r0 for i to 0
        mov   r1, #0          ; set r1 for f to 0
        mov   r2, #0          ; set r2 for g to 0
        mov   r3, #0          ; separate index of arrays
        mov   r4, #10         ; set r4 for constant number 10
        adr   r5, a           ; load r5 with address of base of a array
        adr   r6, b           ; load r6 with address of base of b array
        adr   r7, c           ; load r7 with address of base of c array
        adr   r8, d           ; load r8 with address of base of d array
        adr   r9, e           ; load r8 with address of base of e array
.TEST:
        cmp   r0, r4          ; compare i with number 10
        bge   LOOPEND         ; if i >= 10, exit loop
.BODY:
        ldr   r11, [r5, r3]   ; get value of a[i]
        ldr   r12, [r6, r3]   ; get value of b[i]
        mul   r10, r11,r12    ; a[i] * b[i]
        ldr   r11, [r7, r3]   ; get value of c[i]
        ldr   r12, [r8, r3]   ; get value of d[i]
        mul   r11, r11,r12    ; c[i] * d[i]
        add   r10, r10,r11    ; (a[i] * b[i]) + (c[i] * d[i])
        ldr   r11, [r9, r3]   ; get value of e[i]
        sub   r10, r10, r11   ; (a[i] * b[i]) + (c[i] * d[i]) - e[i]
        add   r1, r1, r10     ; f = f + (a[i] * b[i]) + (c[i] * d[i]) - e[i]
```

```
        ldr     r11, [r5, r3]       ; get value of a[i]
        ldr     r12, [r8, r3]       ; get value of d[i]
        add     r10, r11,r12        ; a[i] + d[i]
        ldr     r11, [r6, r3]       ; get value of b[i]
        ldr     r12, [r7, r3]       ; get value of c[i]
        sub     r11, r11,r12        ; b[i] - c[i]
        mul     r10, r10,r11        ; (a[i]+d[i])*(b[i] – c[i])
        ldr     r11, [r9, r3]       ; get value of e[i]
        mul     r10, r10, r11       ; (a[i]+d[i])*(b[i] – c[i])*e[i]
        add     r2, r2, r10         ; g = g + (a[i]+d[i])*(b[i] – c[i])*e[i]
.UPDATE :
        add     r3, r3, #4          ; add one word offset to array index
        add     r0, r0, #1          ; i = i + 1
        b       .TEST
.LOOPEND:
```

Block	# of instructions	# of cycles
INIT	**10**	**10**
TEST	**2**	**2 best, 4 worst case**
BODY	**20**	**20**
UPDATE	**3**	**3**

Tloop = Tinit + N * (Tbody + Tupdate + Ttest_best) + Ttest_worst
= 10 + 10 * (20 + 3 + 2) + 4
= 10 + 10 * 25 + 4
= 10 + 250 + 4
= 264

10. Are Memory Management Units (MMU) used a lot in Embedded Systems? Which type of Embedded Systems use MMUs the most, today? (10 points)

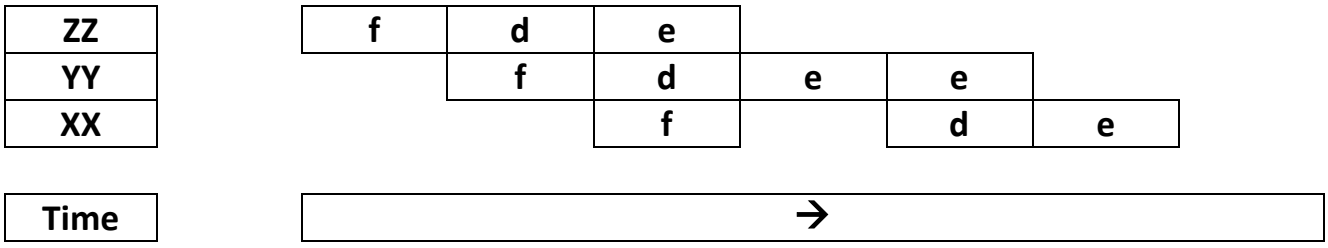
The reason why MMU is not used a lot in Embedded systems is that virtual memory requires a secondary storage device such as a disk. And ARM architecture use MMSs the most these days.

11. An ARM7 has three fictional instructions XX, YY, and ZZ. The YY instruction always requires two cycles to complete the execute stage. Draw the pipeline execution diagram for these three sequence of instructions:

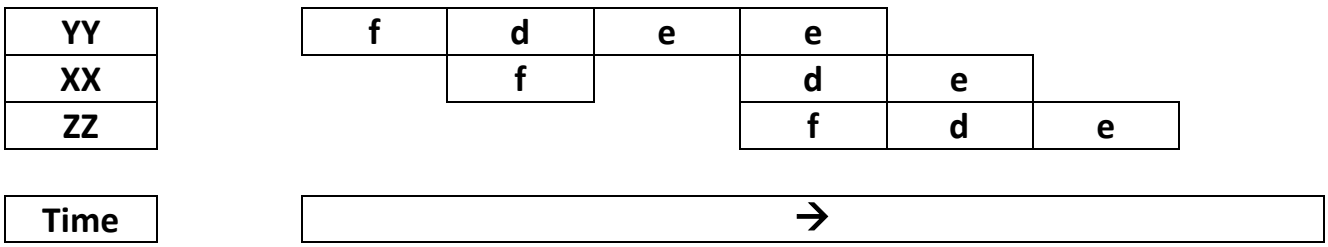
(a) ZZ, YY, XX;
(b) YY, XX, ZZ;
(c) XX, ZZ, YY, XX, ZZ.

(30 points)

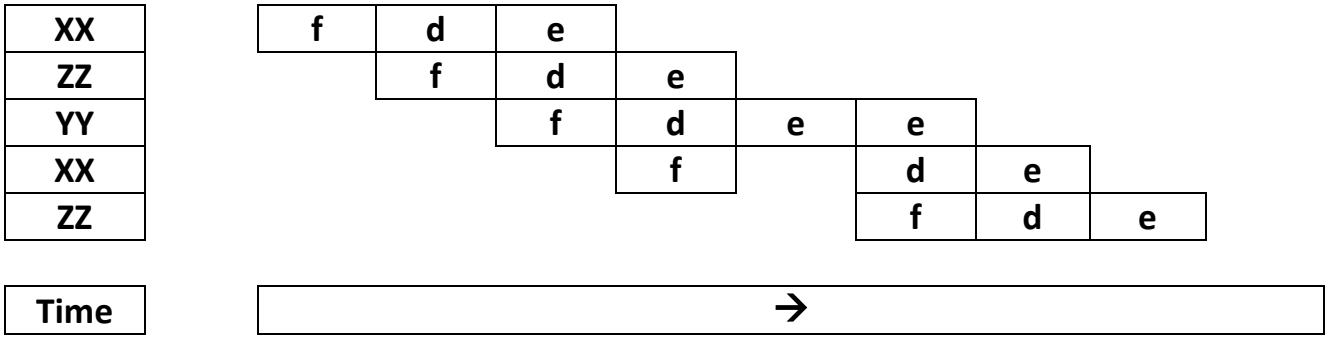
- **(a) ZZ, YY, XX;**
  - **f stands for fetch, d decode, e, execute**



- **(b) YY, XX, ZZ;**
  - **f stands for fetch, d decode, e, execute**



- **(c) XX,ZZ,YY,XX,ZZ;**
  - **f stands for fetch, d decode, e, execute**



12. Which two system's variables increase dynamic power consumption in CMOS circuitry of Embedded Systems? Briefly explain why. (10 points)

$$P = V^2 \cdot C \cdot f$$

V is power supply voltage
C is CMOS circuitry Capacitance
f is clock frequency

Two system's variables are Voltage level and Clock frequency. Because the power consumption of a CMOS circuit is proportional to the square of the power supply voltage and proportional to the clock frequency.

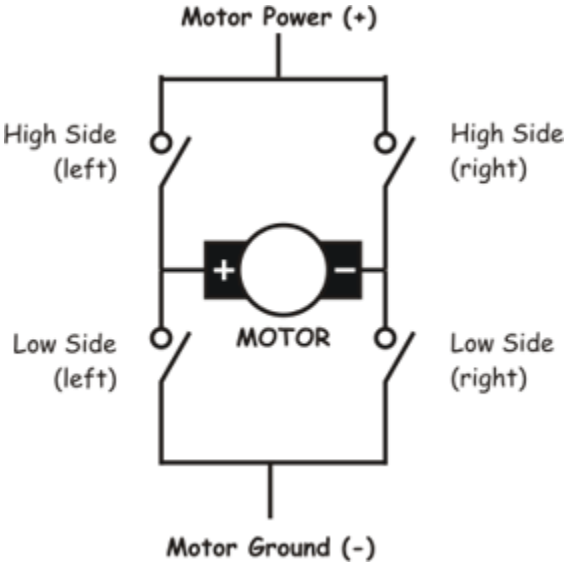
13. What are four power-saving strategies that are used in the CMOS CPUs? (10 points)

1. **Reduce voltage level**
2. **Lower clock frequency**
3. **Disable certain function units that are not required for the currently executing function**
4. **Disconnect from the power supply to eliminate leakage currents**

14. Design of DC motor control using PIC16F microcontroller and H-bridge chip (80 points)

One of the common usage of PIC16F microcontrollers is DC motor control. PIC16F microcontrollers cannot drive directly DC motors because PIC16F microcontrollers usually operate at +5V or +3.3V power supply and its I/O pins can provide only up to 25mA current. Commonly used DC motors require 12V power supply and 300mA current so it is clear that PIC16F cannot drive DC motors directly.

We can use H-bridge which is shown below to drive DC motors.



By applying one 12V signals to connect DC motor "+" side to either 12V motor power or to motor GND and one 12V signal to connect DC motor "-" side to either 12V motor power or to motor GND we can drive the DC motor clockwise or counterclockwise. If both DC motor "+" side and "-" side are connected to 12V or GND at the same time, the DC motor would be halted. We can control the H-bridge with the PIC16F microcontroller to drive a 12V DC motor.

<u>Project specifications</u>

Using PIC16F877A that runs at clock frequency 20MHz and uses 5V power supply connect two 12V DC motors using H-bridge chip L293D and make sure that both 12V DC motor can be driven clockwise, counterclockwise, and that they can be halted by software running on the PIC16F877A microcontroller that check pushbuttons status.

When the Pushbutton 1 is open, the PIC16F877A reads Voltage high and 12V DC motor 1 is halted and when the Pushbutton 1 is closed the PIC16F877A reads Voltage low (GND) and 12V DC motor 1 is running clockwise or counterclockwise depending on the Pushbutton 2 status.

When the Pushbutton 2 is open, the PIC16F877A reads Voltage high and 12V DC motor 1 is driven clockwise and when the Pushbutton 2 is closed, the PIC16F877A reads Voltage low (GND) and 12V DC motor 1 is driven counterclockwise.

When the Pushbutton 3 is open, the PIC16F877A reads Voltage high and 12V DC motor 2 is halted and when the Pushbutton 3 is closed the PIC16F877A reads Voltage low (GND) and 12V DC motor 2 is running clockwise or counterclockwise depending on the Pushbutton 4 status.

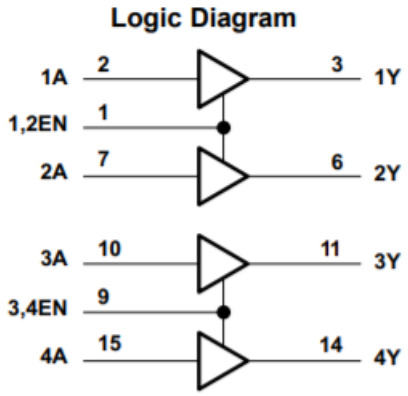
When the Pushbutton 4 is open, the PIC16F877A reads Voltage high and 12V DC motor 2 is driven clockwise and when the Pushbutton 4 is closed, the PIC16F877A reads Voltage low (GND) and 12V DC motor 2 is driven counterclockwise.

Datasheets for PCI16F877A and L293D are provided.

<u>Hardware</u> schematics design is shown below and it needs to be fully connected. You need to do the following:

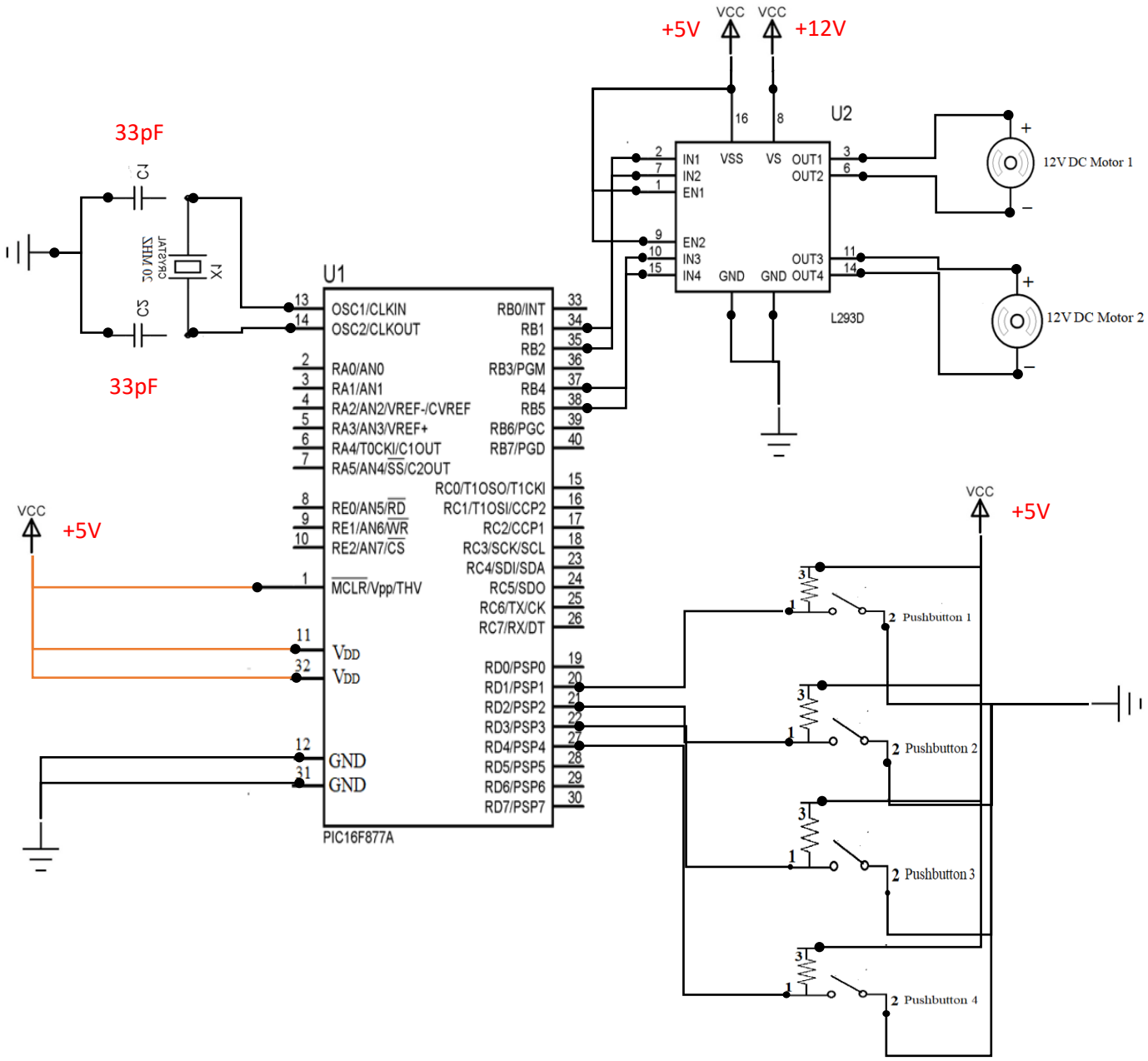
1) **Connect VDD pins to proper voltage supply and GND pins to proper GND on the PIC16F877A microcontroller. Show your full written explanation.**
    a. **#11, #32 pins are Vdd and are connected to +5V and #12, #31 pins are Vss and are connected GND.**
2) **Connect 20MHz clock oscillator to the PIC16F877A microcontroller with proper capacitors values defined. Show your full written explanation.**
    a. **#13, #14 pins are Oscillator input and output each and they are connected to oscillator with 33pF capacitors for 20 MHz clock. And they are also connected to GND as well.**
3) **Pin 1 on the PIC16F877A needs to be connected for proper work of the PIC16F877A microcontroller. Show your full written explanation.**
    a. **#1 pin is connected to +5V because it is the master clear pin of this IC. It resets the microcontroller and is active low, meaning that it should constantly be given a voltage of 5V and if 0 V are given then the controller is reset.**
4) **Connect Pushbutton 1 thru 4 pins 1, 2, and 3 to proper VDD, GND and pins on the PIC16F877A. Use PORTD on the PIC16F877A to connect Pushbuttons 1 thru 4. Show your full written explanation.**
    a. **#20, #21, #22, #27 pins are connected to Pushbutton 1,2,3,4 each. They also need to be connected with +5V and GND. Pushbuttons also need to be connected +5V and GND.**
5) **Use PORTB on the PIC16F877A to connect it to L293D H-bridge chip. Show your full written explanation.**

a. **#34, #35, #37, #38 pins are PORTB's Input and output. They are connected to L293D's #2, #7, #10, #15 pins. Just like the image below(From the L293 datasheet)**



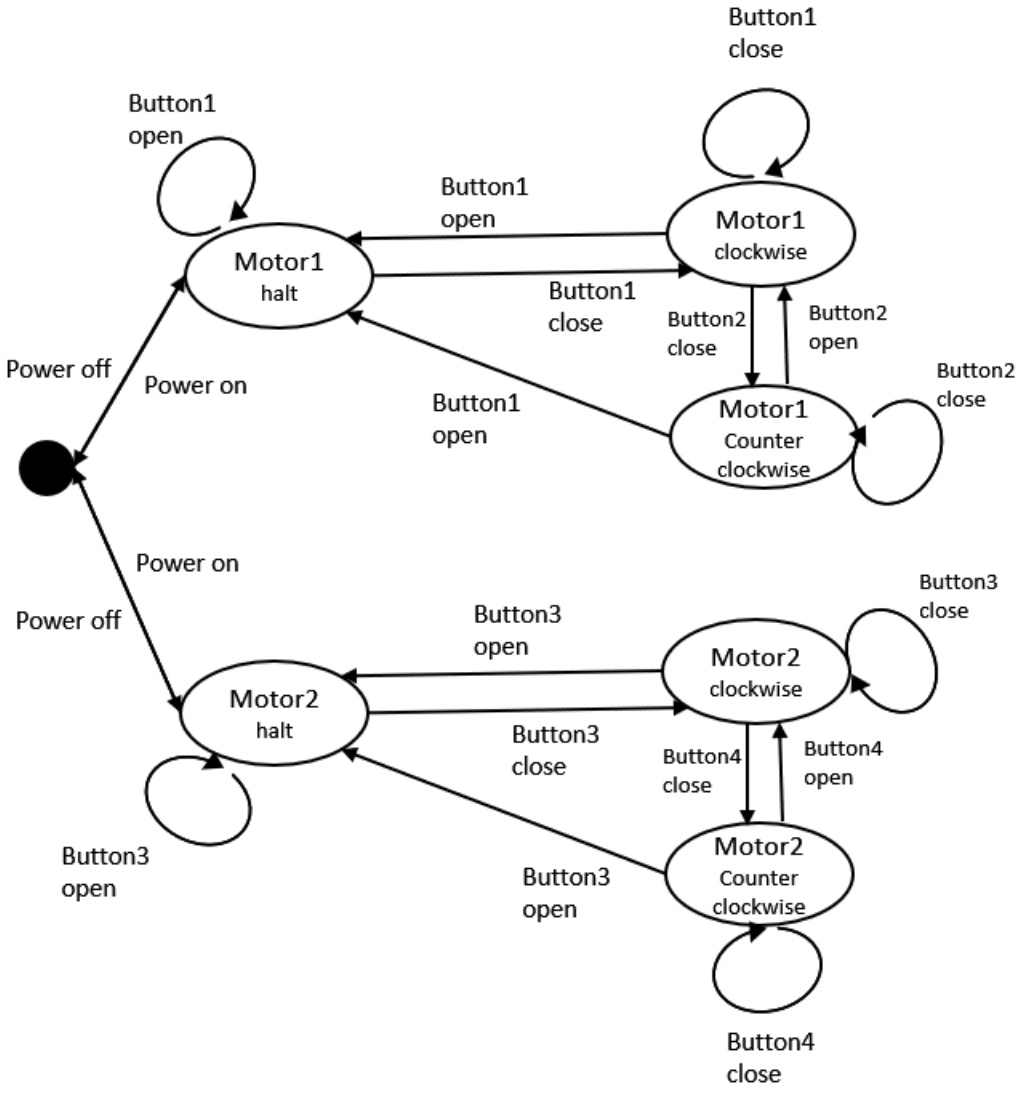
6) **Connect all pins on the L293D H-bridge chip to all proper voltages, GND, proper interface pins from the PIC16F877A and to 12V DC motors "+" and "-" pins. Show your full written explanation.**
   a. **First of all, #16 is connected to +5V and #8 is connected to +12V. Because DC motor will work on 12V so #8 pin is connected to +12V and this will power DC motors. And #4, #5, #12, #13 pins are connected to GND. And then #1 and #9 pins are connected to +5V because they can enable driver channels. Then #3, #6 pins are connected to 12V DC Motor 1 and #11, #14 pins are connected to Motor 2.**

Byeongchan Gwak, Student ID : 501026



<u>Software</u> design should include the following:

1) **The state machine that runs an infinite loop and shows all possible states for the given hardware design.**



2) **What programming language and what complier would you use to write the code for this design? Give a reasonable explanation based on what you know about PIC16F877A microcontroller.**
   a. **I rather use C and C compiler. Because developing code using C frees programmers from the details of multi-byte math and paging, and generally improves code readability and maintainability.**

15. Design of LED light intensity control using PIC16F microcontroller and Digital-to-Analog Converter (DAC) chip (50 points)

Another common usage of the PIC16F microcontrollers is interface to Analog-to-digital converters (ADC) and interface to Digital-to-Analog converters (DAC) for various signals. This design has requirement to connect 12-bit, 5V DAC chip MCP4921 to the PIC16F877A microcontroller. By adjusting the output voltage of the DAC chip from 0V to 5V, the LED light intensity is controlled.

The requirements are the following:

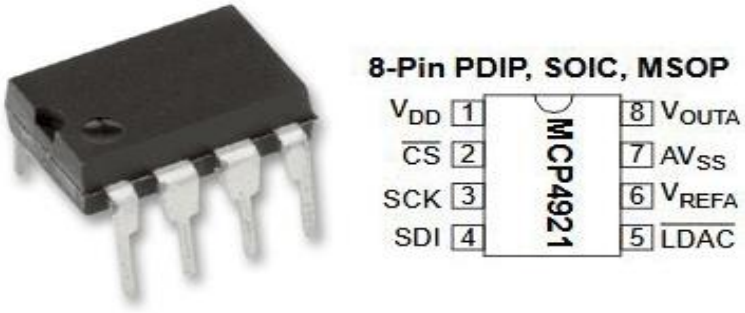
(1) The PIC16F877A microcontroller has to work on 20MHz and has 5V power supply voltage. On the circuit Schematics, connect properly 20MHz crystal oscillator and proper Capacitors values to the proper pins on the PIC16F877A microcontroller. Show your full written explanation.

- **a. #13, #14 pins are Oscillator input and output each and they are connected to oscillator with 33pF capacitors for 20 MHz clock. And they are also connected to GND as well.**

(2) On the circuit Schematics, connect properly pins 11, 12, 31, 32 and 1 since the PIC16F877A has to work at 5V. Show your full written explanation.

- **a. #11, #32 pins are Vdd and are connected to +5V and #12, #31 pins are Vss and are connected GND. #1 pin is connected to +5V because it is the master clear pin of this IC.**

(3) 12-bit DAC chip MCP4921 has to work at 5V and has to be interfaced to the PIC16F877A microcontroller using serial bus SPI interface which is part of the both chips. MCP4921 pinout is shown below.



MCP4921 Pin No.	MCP4922 Pin No.	Symbol	Function
1	1	V_{DD}	Positive Power Supply Input (2.7V to 5.5V)
—	2	NC	No Connection
2	3	$\overline{CS}$	Chip Select Input
3	4	SCK	Serial Clock Input
4	5	SDI	Serial Data Input
—	6	NC	No Connection
—	7	NC	No Connection
5	8	$\overline{LDAC}$	Syncronization input used to transfer DAC settings from serial latches to the output latches.
—	9	$\overline{SHDN}$	Hardware Shutdown Input
—	10	V_{OUTB}	DAC_B Output
—	11	V_{REFB}	DAC_B Voltage Input (AV_{SS} to V_{DD})
7	12	AV_{SS}	Analog ground
6	13	V_{REFA}	DAC_A Voltage Input (AV_{SS} to V_{DD})
8	14	V_{OUTA}	DAC_A Output

On the Schematics connect all pins of the 12-bit, 5V DAC chip. The DAC chip is used for 5V output voltage signal with 12 bit resolution. What is 5V output voltage signal resolution if DAC is 12-bit chip? Show your full written explanation.

Datasheets for PCI16F877A and MCP4921 are provided.

a. **#1, #6 pins of the MCP4921 are connect to +5V and #5, #7 are also connected to GND. #8 pin is connected to LED because it generates DAC output. According to MCP4921 datasheet, three input signals are required. Therefore, #2(CS), #3(SCK) #4(SDI) pins are connected to #24, #25, #26 each. MCP4921 is a 12 bit DAC, so MCP4921 will provide 12 bits of output. For 12-bit, it is = 4096. This means 12-bit resolution DAC could produce 4096 different outputs.**

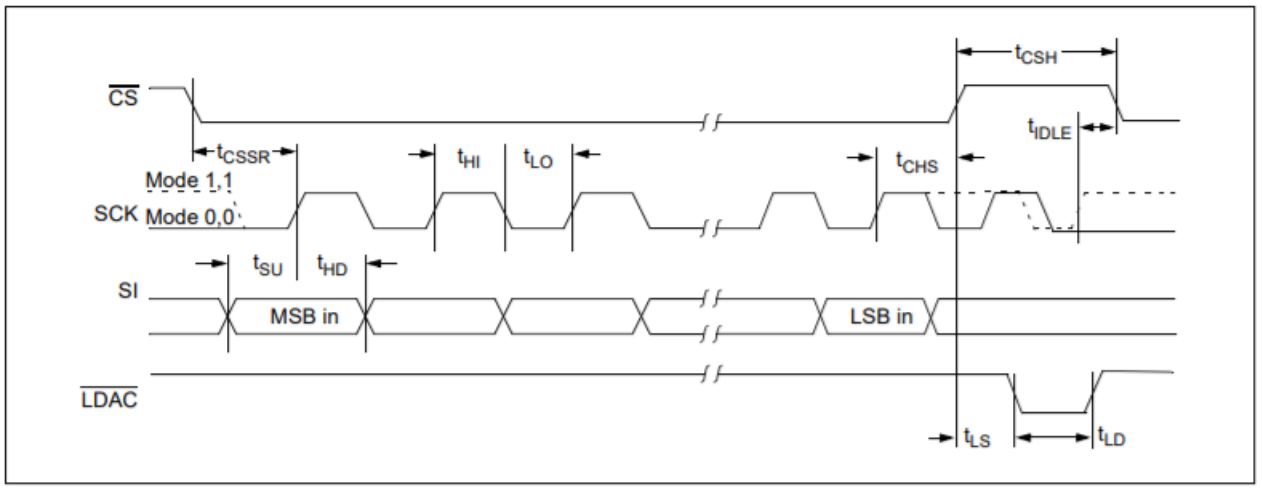


FIGURE 1-1: *SPI Input Timing Data.*

(4) Connect properly pins 1 and 2 of the LED circuit. Show your full explanation.

a. #1 pin of LED is connected to #8 pin of the MCP4921 and #2 pin of LED is also connected to GND. #8 pin of the MCP4921 generate output value and LED light intensity will follow the output of MCP4921.