EECS388 Embedded Systems Fall 2022

HW 1 – Solution

Part 1 – True/False (1 point each)

Indicate true or false for each of the following statements:

1. Assembly code is generated before the machine code binary in a compiler toolchain. (T/F)

Ans: True.

2. You can manually perform everything that a makefile does. (T/F)

Ans: True.

3. Makefiles are bad for maintainability and portability compared to IDEs. (T/F)

Ans: False.

4. `Volatile` is a C type qualifier that provides specific instructions to the compiler on how the variables should be managed. (T/F)

Ans: True.

5. In 2's complement binary representation, the most significant bit is the sign bit. (T/F)

Ans: True

6. Each general-purpose computer has its own customized ISA. (T/F)

Ans: False.

7. Operands in an instruction hold input/output data. (T/F)

Ans: True.

8. A memory with higher addressability has a higher capacity. (T/F)

Ans: False.

9. General-purpose registers hold opcode of an instruction. (T/F)

Ans: False.

10. Components of a Von Neumann computer are Input/Output, Memory, Processing Unit, and Controller. (T/F)

Ans: True.

11. A Von Neumann computer is a little-endian computer. (T/F)

Ans: False.

12. In a Von Neumann computer model,

    a. IR and PC are part of the control unit. (T/F)
       Ans: True

    b. the next value of PC depends on the current value of IR. (T/F)
       Ans: True
       E.g.: JMP or BR change PC

    c. programs are stored in memory. (T/F)
       Ans: True

    d. the register file is part of the controller. (T/F)
       Ans: False

    e. controller decodes the instructions. (T/F)
       Ans: True

13. We have three types of instructions in LC-3: Operate, Date movement, and Control instructions. (T/F)

Ans: True.

14. We need 3 bits to address each register in a register file with 8 registers. (T/F)

Ans: True.

15. Every instruction in LC-3 needs to go through the FETCH phase of instruction processing. (T/F)

Ans: True.

16. Every instruction in LC-3 needs to go through the EVALUATE ADDRESS phase of instruction processing. (T/F)

Ans: False.

E.g.: Operate instructions.

17. In PC-relative addressing mode, the processor needs to perform one memory access to evaluate the final address. (T/F)

Ans: False.

18. Control instructions alter the sequential execution of instructions. (T/F)

Ans: True.

E.g.: JMP, BR, JSR...

Part 2 – Multiple Choice (2 points each)

Select the appropriate options for the following questions:

1. What is a cyber-physical system?
    a. A computer system that runs a specific application
    b. Another name for embedded systems
    c. A physical system that includes a computer inside
    d. All above

Ans: D

2. What are the differences between a general-purpose computer and an embedded system?
    a. Embedded systems only run one application, but general-purpose computers run many applications.
    b. Embedded systems are not programmable, but general-purpose computers can be programmed.
    c. Embedded systems usually have runtime constraints but for general-purpose computers, faster execution is always better.
    d. All above.

Ans: D

3. Which one is correct regarding the efficiency of embedded systems?
    a. Since embedded systems usually have power constraints, then it is important to reduce their power consumption.
    b. Embedded systems need to be optimized for size, power, weight, and cost.
    c. Performance and efficiency are equally important for embedded systems.
    d. All above.

Ans: D

4. Which one of the following variables can represent the decimal range of $[2^{32} -1, -2^{32}]$? (Assuming `int` type is 32 bits)
    a. int var;
    b. long int var;
    c. long long int var;

d. unsigned int var;

Ans: both B and C

If it was [2^{31} –1, -2^{31}] then, the answer also include "int var".

5. What does code (1) and (2) print after executing the main function?
   a. 10, 11
   b. 10, 10
   c. 11, 11
   d. 11, 10

```
// Code 1
void add(int a) {
    a = a + 1;
}

void main()
{
   int var = 10;
   add(var);
   printf("%d\n", var);
}
```

```
// Code 2
void add(int *a) {
    *a = *a + 1;
}

void main()
{
   int var = 10;
   add(&var);
   printf("%d\n", var);
}
```

Ans: A Code 1 prints 10 and Code 2 prints 11.

6. What does the following code print?

   a. 10, <Address of `var`>
   b. <Address of `var`>, 10
   c. 10, 10
   d. The code has errors

```
void main()
{
   int *var;
   …
   *var = 10;
   printf("%d, %d\n", var, *var);
}
```

Ans: If you execute the code in its current format you get a segmentation fault (Option D). But since we did not discuss memory

allocation in class, you get points if you also select option B (assuming that in the ellipsis there is a malloc statement).

7. When can we expect that the PC value does not change in LC-3?
    a. After executing an Operate instruction
    b. After executing a Data Movement instruction
    c. After executing a Control instruction
    d. None of the above

Ans: C

E.g., in the following code when we execute the JMP instruction the PC does not change:

```
        .ORIG  0x3000
LABEL   JMP    LABEL
```

8. Consider a Load instruction with PC-relative addressing mode. What is the range of addresses that the instruction can load from if the width of PC offset is 5 bits? Note that PC offset is a 2's complement number.
    a. [PC + 15, PC - 16]
    b. [PC + 16, PC - 15]
    c. [PC + 32, PC - 31]
    d. [PC + 31, PC - 32]

Ans: A

With 5 bits we can address

$[-2^4, 2^4 - 1]$ -> [15, -16]

In general, with an n-bit 2's complement number, you can address the $[-2^{n-1}, 2^{n-1} - 1]$ range.

Part 3 – Short Answer (2 points each)

Provide a short answer to the following questions:

9. Why is a native compilation not usually an option for embedded systems?

Ans: Because of the limited resources.

10. Why do we need a makefile? Provide three reasons.

Ans: 1) Reduce Error.
2) Improve Scalability.
3) Its faster than a manual build.

11. Write two reasons for using a version control software such as git.

Ans: 1) Facilitate Collaboration.
2) Track Changes.

12. Why do we learn C in EECS388?

Ans: C is the dominant programming language in Embedded systems.

13. Write a C statement that allocates 8 bits in memory. (Single line of code)

Ans: char var;

14. What is the result of the following bitwise operations?

(signed) 10011011 << 2 =
(unsigned) 10011011 << 2 =

(signed) 10011011 >> 2 =
(unsigned) 10011011 >> 2 =
~ (signed) 10011011 =

Ans: 011 011 00
011 011 00
111 001 10
001 001 10
011 001 00

15. In C, a function cannot return multiple variables. What is a solution for modifying several input variables inside a C function?

Ans: Passing by pointer.

16. What are the two components of an LC-3 instruction? What info each of these components contains?

Ans: 1) Operand -> data to be used by the instruction.
2) Opcode -> operation to be performed.

17. The following is the machine code for an LC-3 instruction:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1

Answer the following question based on the LC-3 instruction format table.

a. Opcode is ……
b. Mnemonics is ……
c. (True/False) This instruction updates condition codes after execution. (T/F)
d. (Multiple choice) Bits [2:0] of the instruction represents
    i. Source Register
    ii. Destination Register

iii. Immediate value
iv. Opcode

e. (True/False) This instruction updates the register file after execution.

Ans: a) 0b 0001
b) ADD
c) True
d) Source register bit [5] ==0
e) True

18. What is the purpose of condition codes in LC-3?

Ans: It enables conditional branches.

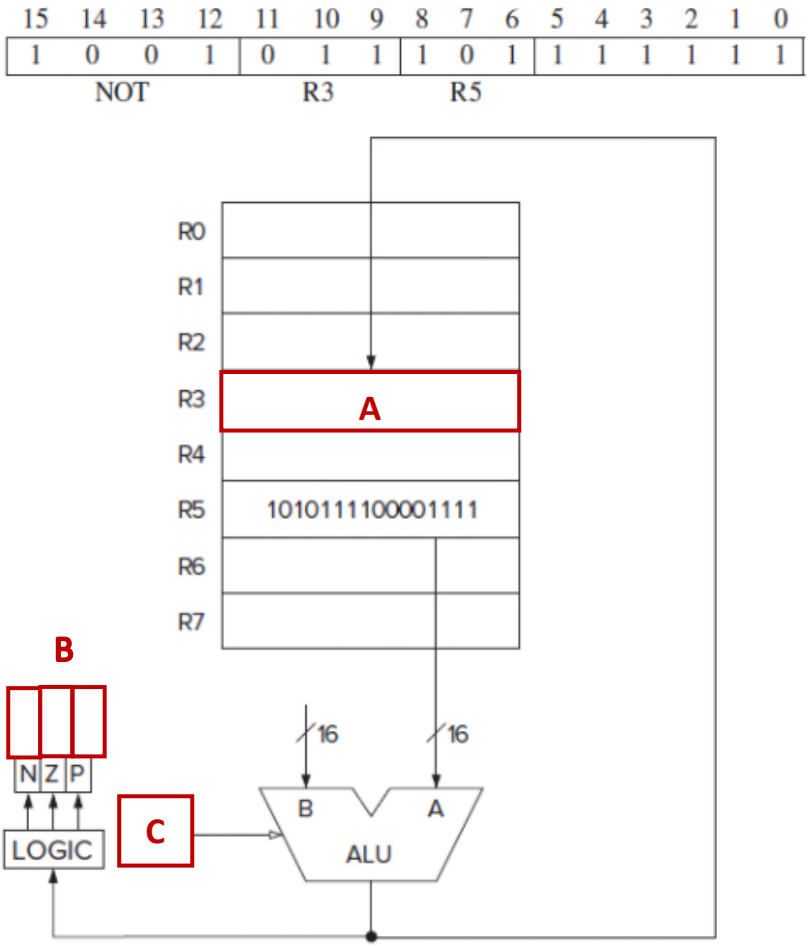
19. What is the value of the condition code registers (N, Z, P) after executing the following piece of code in LC-3?

```
AND   R1,R1, 0x00
NOT   R2,R1
```

Ans: 0x0000 => The result of NOT is 0xFFFF

0xFFFF => N = 1, Z = 0, P = 0

20. Assuming the following instruction is being executed in LC-3, fill in the blanks labeled as “A”, “B”, “C”.



Ans: A) 0101000011110000

B) 001

C) NOT

21. What is the purpose of "TRAP" instruction?

Ans: It requests service from the OS.

Part 4 – Other Questions (3 points each)

1. Write the LC-3 assembly representation of the following C code. Assume R1, R2, R3, R4, R5, R6 are in R1, R2, R3, R4, R5, R6 registers, respectively.

Ans:

```
if (R5 == R6)
      R1 = R2 + R3;
else
      R1 = R3 + R3;
R4 = R1 + R1;
```

What we need to do is to first use a sequence of Operate instructions to set the condition codes to a value which we can use to implement R5 == R6 condition. The easiest way is to subtract R5 from R6, if the result it zero (i.e., condition code is [Z=1, N=0, P=0]), then the condition of the if statement is true, otherwise it is false. Since we don't have SUB instruction in LC3, we need to use 3 instructions to implement SUB (refer to Slide#14-Lecture-5). The following is one way to implement the above C-code in LC3:

	NOT	R7, R6
	ADD	R7, R7, #1
	ADD	R7, R7, R5
	BRz	IF
	ADD	R1, R3, R3
	JMP	FINISH
IF	ADD	R1, R2, R3
FINISH	ADD	R4, R1, R1

2. Assume a 16-bit instruction with the following format:

OPCODE	DR	SR	IMM
4 Bits	5 Bits	5 Bits	

If there are 12 opcodes and 32 registers, what is the maximum number of bits that we can assign to the immediate field (IMM):

Ans:

- We need 4 bits to represent 12 opcodes
- We need 5 bits to address 32 registers

=> 16 – 4 – 5 – 5 = 2 bits remains for IMM

3. State different phases of executing an instruction. Briefly explain what operations take place in each phase.

Ans: 1) Fetch: Fetch the next instruction pointed by PC from the memory and increment PC

2) Decode: Decode the instruction type and generate appropriate control signals

3) Evaluate Address: Generate the memory address for accessing the memory

4) Fetch Operands: Read operands from memory or reg file

5) Execute: Execute the operation in ALU

6) Write results: Write results in the reg file or memory

4. Consider the following memory organization

Addr	Data (1 byte)
0x0	
0x1	
0x2	
0x3	
0x4	
0x5	
0x6	
0x7	

a. What is the addressability and address space of this memory?

Ans: Addressability = 1 byte

Address space = 8 unique addresses

b. Assuming that the memory is Little Endian and is initialized with 0s, what is the final content of the memory after storing the following multi-byte variable in the memory:

```
int short var = 0x5678;
assume &var -> 0x0;
```

Ans:

0x78
0x56
0x00
0x00
. . .
0x00

5. Fill in the table:

Decimal	2's complement?	Binary	Hexadecimal
0	NO	0b0	0x0
2	Yes	0b010	0x2
9	Yes	0b01001	0x09
-128	Yes	0b1000 0000	0x80
131	No	0b1000 0011	0x83
-32768	Yes	0b1000 0000 0000 0000	0x8000

EECS388 Embedded Systems Fall 2022

HW 2 – Due date: Friday Dec 9th **11:59PM**

Canvas submission.

****** Important: There is no late policy. It means that if you submit after the deadline you'll receive a zero! ******

The reason is that we release the solution right after the deadline

Part 1 – True/False (1 point each)

Indicate true or false for each of the following statements:

T 1. Function (subroutine) implementation requires Control instructions. (T/F)

T 2. We have one register file in LC-3 and all the instructions in all subroutines read and write from that one register file.

T 3. We use regular load and store instructions to access the status and data registers of a Memory Mapped IO (MMIO) device.

T 4. We use a handshaking protocol to control the exchange of data between a microcontroller and an asynchronous I/O device.

F 5. In a system with an OS, a programmer can directly access I/O registers.

T 6. The TRAP instruction in LC3 requests service from the OS running in the privilege mode.

F 7. TRAP Vector Table is a component inside the Processing Unit in the Von Neumann computer model.

F 8. Interrupt and Trap are the same thing.

F 9. An external interrupt can always preempt the execution of the currently running program on the CPU.

F 10. The Interrupt enable-bit ensures that the highest priority interrupt is always selected to be sent to the CPU.

T 11. In case several devices request to send an interrupt to the CPU simultaneously, the highest priority device is selected to send the Interrupt.

T 12. An interrupt is sent to the processor whenever a device needs attention.
you can interprete this as "false" as well if you think about pri

T 13. In direct mode interrupt, we have a common interrupt handler that finds the cause of the received interrupts.

T 14. An I/O interface is considered synchronous if there is a clock signal between the sender and receiver.

F 15. We cannot have a parallel and synchronous interface.

F 16. RS232 is a synchronous serial protocol.

F 17. I2C uses a shift register to shift out bits to the output port and shift in bits from the input pin for communication.

T 18. PWM is an effective way of controlling DC motors' speed, because the speed of a DC motor changes with the average input voltage.

F 19. Brushed motors have a permanent-magnet rotor.

T 20. PWM is a simple way to generate signals with various average voltages.

T 21. We often use PWM to control the angle and position of servo motors.

F 22. Stepper motors use PWM.

F 23. Stepper motors are equipped with a positional feedback device.

T 24. Baremetal execution is faster than OS-assisted execution (assuming similar hardware config).

T 25. Process Control Block (PCB) data structure is used to support multiple active processes in a system.

T 26. Context switches have overhead.

F 27. A real-time scheduler should always prioritize more critical tasks.

T 28. The priority of jobs dynamically changes in an EDF scheduler.

Question #2: Regarding option "d", you can argue that a fast device that does not receive frequent events does not benefit from polling. Think about a high bandwidth network adapter that does not receive any packet ….
If you connect speed to the frequency of events, then option "d" is correct.

Part 2 – Multiple Choice

Select the appropriate options for the following questions:

1. Why do we need to save *PC* before a subroutine call? [2 points]
   a. To know the return address
   b. To be able to perform operations on the PC register in the subroutine
   c. Not to miss any of the register values saved in the register file
   d. To use it and jump to the starting address of the subroutine

2. Select all correct statements about polling and Interrupt (chose all the correct statements) [+1 point for each correct selection]
   a. Polling or Interrupt are two mechanisms that are used to identify whether an I/O device has data (or need attention)
   b. A device with bursty behavior can adaptively use polling and interrupt to reduce Interrupt's context switches and Polling's CPU cycle waste.
   c. Polling save CPU resources
   d. Polling is better for fast devices and Interrupt is better for slow devices.
   e. We used the Interrupt mechanism when implementing blinky LED in the lab. We used a timer and interrupt
   f. We can implement Polling by checking the value of a volatile MMIO register in a while loop.
   g. An Interrupt can result in the preemption of the currently running program.

3. What is the benefit of TRAP (system call)? [2 points]
   a. It implements protection for shared device registers
   b. Improves programmers productivity
   c. Hides low-level details of I/O device communication from the programmer.
   d. All above

4. What is a *TRAP service routine*? [2 points]
   a. A function that executes in the privileged mode on behalf of the user
   b. A function that executes by the OS and provides a service to the user
   c. A function that its instructions are placed in the user memory space.
   d. All above.
   e. A & B

5. What is the functionality of the *TRAP Vector Table*? [2 points]
   a. Store the starting address of TRAP service routines
   b. Store the starting address of interrupt service routines
   c. Store the first instruction of the TRAP service routines
   d. Store the return address to the next instruction after the TRAP instruction

6. The advantage of serial interfaces is that: [2 points]
   a. They use fewer pin on the processor and I/O device
   b. They consume lesser power
   c. They deliver higher bandwidth (bit rate) compared with a parallel interface.
   d. All above
   e. A & B

7. What is stored inside an *interrupt vector table*? [2 points]
   a. Interrupt vector table offset
   b. Address of an interrupt service routine
   c. Interrupt service routine.
   d. Interrupt vector number.

Part 3 – Short Answer

Provide a short answer to the following questions:

1. Assume that we have a keyboard that is connected to a microcontroller. When is operating the keyboard in interrupt-driven mode beneficial? Explain each selection. (multi-choice) [4 points]
   a. The inter-arrival time of keystrokes is fixed.

   b. The inter-arrival time of keystrokes is unknown.

   we don't waste CPU Cycles

   c. The average inter-arrival time of keystrokes is in the same order as the micro controller's frequency

   d. The average inter-arrival time of keystrokes is orders of magnitudes larger than the micro controller's frequency

The interrupt overhead would not be significant since we have few interrupts and we don't waste CPU cycles for polling

2. Why should we save registers in the memory before calling a subroutine or initiating an interrupt? [2 points]

To protect them from being over written by the subroutine and lose data.

3. Why do we need to have a privileged mode of execution in processors? [2 points]

protect share data and resources

4. Why do we set/reset the privileged bit while executing TRAP/RTI instructions? [2 points]

because trap executes in the privilege mode

5. Why do processors handle interrupts in the instruction boundaries? [2 points]

simplicity

6. Why do we save PC and PSR before initiating an interrupt? [2 points]

know where to return after finishing the interrupt service routine

7. Where do we save PC and PSR before initiating an interrupt? [2 points]

Memory (system stack)

8. What is baudrate in UART? [2 points]

The agreed upon speed if the transmission of one bit between sender and receiver

9. Write an advantage and a disadvantage for bus compared with a point-to-point network. [2 points]

Bus:
+ less resources
-Need arbitration

Point to point:
+ faster
- More expensive

10. Your classmate wrote this code and claimed that this would set the content of memory address Value1 to zero. Do you agree with your classmate? What is your proposed solution? [3 points]

NO! R1 is overwritten in FUNC

save R1 & restore it.

```
            .ORIG x3000
(1)         AND     R1, R0, #0 // R1 <- R0 & 0
            JSR     FUNC
(2)         ST      R1, Value1 // Mem[Value1] <- R1
            HALT

FUNC        Add     R1, R1, #1 // R1 <- R1 + 1
            JMP     R7

Value1  (3) .BLKW   1
            .END
```

(1) (2) ST R1, SAVE
LD R1, SAVE

(3) SAVE .BLKW 1

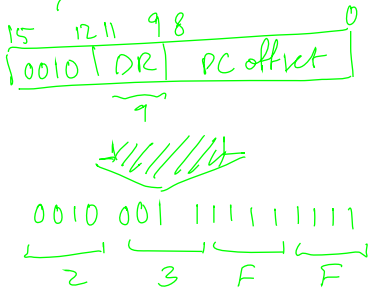
11. Consider the following address space and the following assembly program and answer the questions. (X, Y, Z)

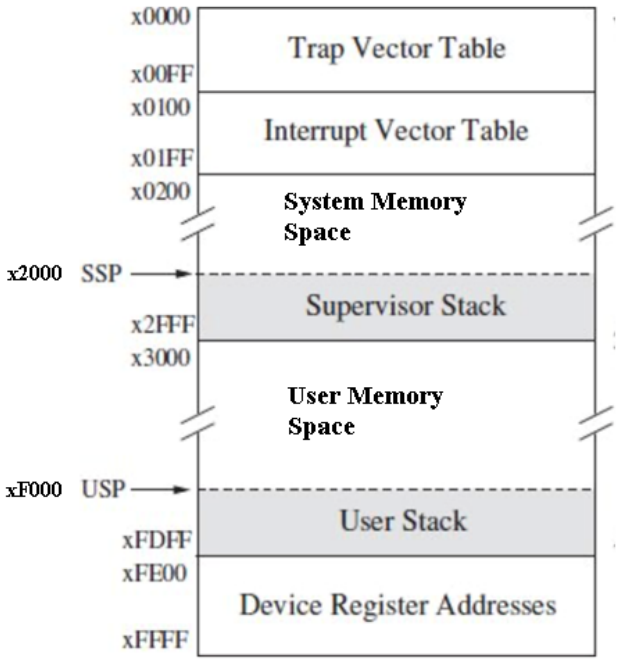
memory address

		Condition Code (N,P,Z)			
x3000	1		.ORIG	x3000	
x3001	2		LDI	R3, L1	// R3 = Mem[Mem[L1]] = Mem[xFE04]
x3002	3		LDI	R4, L2	// R4 = Mem[x2100]
x3003	4		LDI	R5, L3	// R5 = Mem[x0001]
x3004	5		LDI	R6, L4	// R6 = Mem[x02FF]
x3005	6	(0,1,0)	LD	R7, L3	// R7 = Mem[L3] = Mem[x3014] = x0001
x3006	7	(0,1,0)	ADD	R7, R7, #1	// R7 = R7+1 = 0x0002
x3007	8	unchanged	ST	R7, L5	// Mem[L5] = R7 => Mem[x3013] = x0002
x3008	9	"	BRz → not taken	FINISH	
x3009	10	L0	LD	R1, L0	// R7 = instruction "LD R7, L0" = 0x23F
x300A	11		AND	R2, R2, x0000	
x300B	12		ST	R2, L0	
x300C	13	STR	.STRINGZ	"Hi!"	// 'H' // 'i' // '!' // 'NULL'
x300D					
x300E					
x300F	14	L1	.FILL	xFE04	
x3010	15	L2	.FILL	x2100	
x3011	16	L3	.FILL	x0001	
x3012	17	L4	.FILL	X02FF	
x3013	18	L5	.BLKW	2	
x3014	19	FINISH	.END		

lecture 8 slide 8

"LD"





a. Which lines of the code access a Memory-Mapped I/O device? What are the addresses of the IO registers? [2 points]

line 2 0xFE04

b. In which lines do we access a privileged memory space? [2 points]

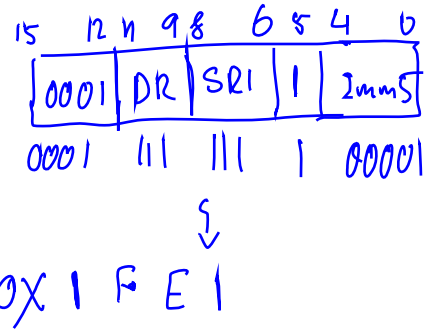
lines 2, 3, 4, 5

c. Do we access Trap Vector Table in the code snippet? [2 points]

Yes, in line 4

d. After the code execution, what is the content of the following memory addresses? Write it in Hex. [4 points]

1. L5: 0x0002
2. L5+1: Unknown !
3. x3005: "ADD R7, R7, #1" ⇒



4. x3008: 0x 0000

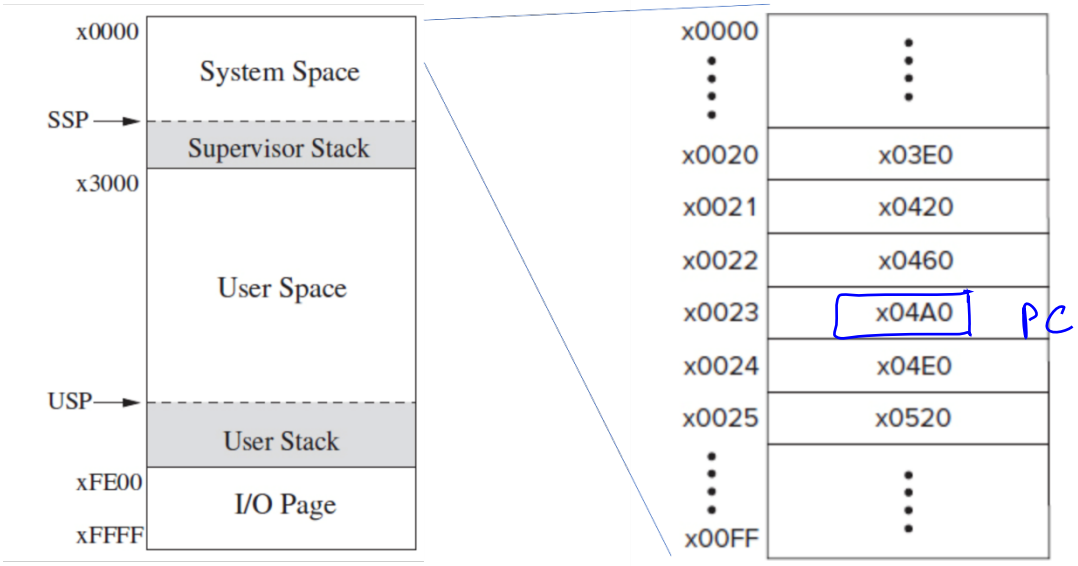
The STORE instruction at line 12 updates address 0x3008 to 0x0000

5. x300B: 0x 0048 → ZEXT (0x 48)

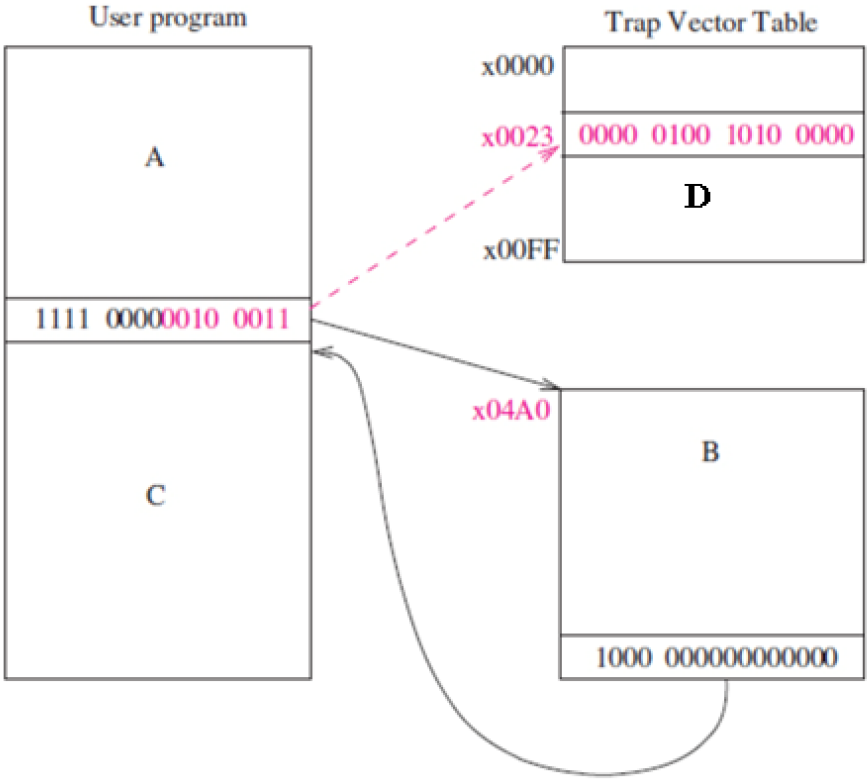
Ascii of "H"

12. Consider the following TRAP vector table in LC3.
What is the PC value after executing the following TRAP instruction? [3 points]

```
.ORIG x3000
TRAP x23
```



13. Answer the following questions based on the figure.



a. What is the trap vector number? [1 points] 0X23

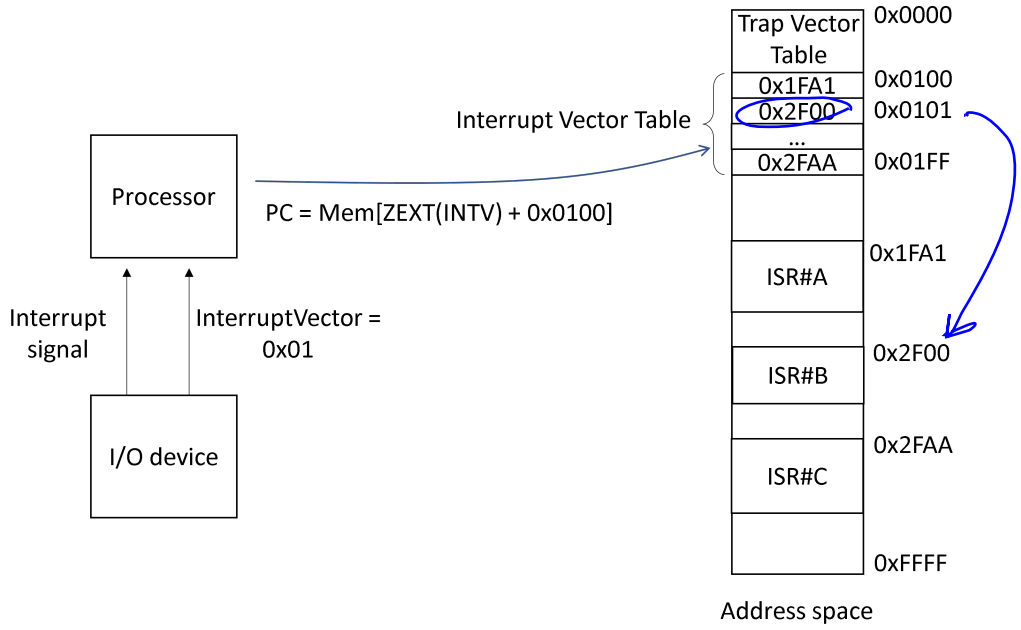
b. What is the size of the Trap Vector Table? [1 points]

256

c. Which section of the memory stores the Trap service routine? [2 points]

a. A
b. B
c. C
d. D

14. Consider the following illustration of a system implementing vectored interrupts. Which ISR will be executed assuming that the interrupt vector number provided by the I/O device is 0x01? [3 points]
    a. ISR#A
    b. ISR#B
    c. ISR#C



15. What are the data that the following bit stream is transmitting? Assume RS-232 serial protocol.Is there any error in the transmission? [4 points]

11111110111111111111111111110000000000011111111

Start Data Stop P

Start data Stop P

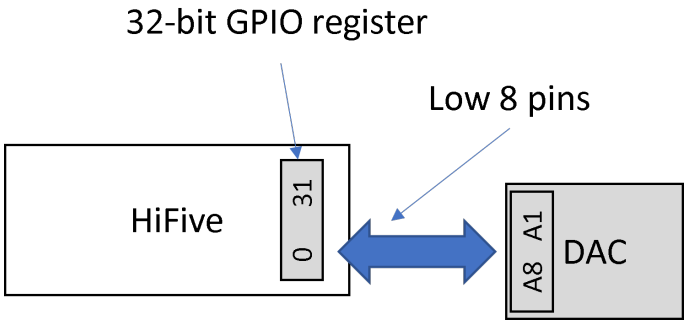
0XFF

has Error because parity should be 0

0x00

No Error.

16. Complete the following code to set the output voltage to 40/256 volts. Assume that the DAC is a +10V output DAC. [4 points]




```
void set_dac()

{

   uint32_t val = *(volatile uint32_t *) (GPIO_CTRL_ADDR +
GPIO_OUTPUT_VAL);

      // YOUR CODE HERE

   *(volatile uint32_t *) (GPIO_CTRL_ADDR + GPIO_OUTPUT_VAL) = val;

}
```

$$V_0 = 10 * \left(\frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \frac{A_4}{16} + \frac{A_5}{32} + \frac{A_6}{64} + \frac{A_7}{128} + \frac{A_8}{256} \right)$$

$$V_0 = \frac{40}{256}$$

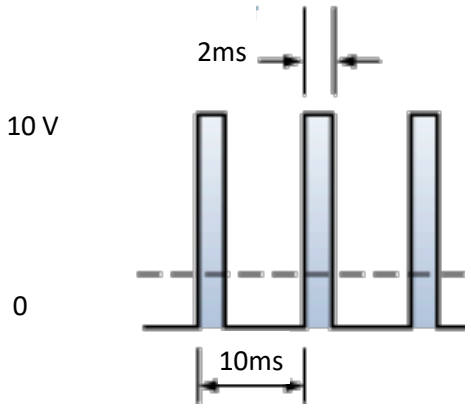
$$\Rightarrow \left(\frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \frac{A_4}{16} + \frac{A_5}{32} + \frac{A_6}{64} + \frac{A_7}{128} + \frac{A_8}{256} \right) = \frac{1}{64}$$

$$\Rightarrow \boxed{A_6 = 1}$$

Code: val &= 0XFF FF FF 00;

val |= 0X 00 00 00 04;

17. What is the average voltage of the following PWM signal? [2 points]



Duty cycle = $\frac{2ms}{10ms} = 0.2$

Avg voltage = $10 * 0.2 = 2V$

EECS388 Embedded Systems
Practice Questions

True/ False:

1. L&L bound can prove that a task set is not schedulable with RMS.
   False.
2. The exact schedulability test calculates the total interference caused by higher priority tasks for a lower priority task.
   True.
3. It is impossible to have priority inversion in a system with two tasks.
   True
4. Priority ceiling protocol prevents certain deadlocks.
   True

Short Answer:

1. Implement a subroutine that multiplies the value of memory stored in Label1 by the value stored in Label1.

```
            .ORIG x3000
            JSR   MUL
            HALT
MUL         // your code here
            // R3 = Mem[Label1] * Mem[Label2]
Label1      .BLKW 1
Label2      .BLKW 1
            .END
```

```
            LD          R1, Label1
            LD          R2, Lable2
LOOP        BRz         DONE
            ADD         R1, R1, R1
            ADD         R2, #-1
            JMP         LOOP
DONE        RET
```

Sine we don't use R1 and R2 in the main function, we don't need to save or restore them.

2. What are the data that the following bit stream is transmitting? Assume RS-232 serial protocol.Is there any error in the transmission?

```
11111110110101111111111111000011000011111111
```

```
1111111 0(start) 11010111 (first data reverse) 1 (p)
11 (stop) 111111111 0 (start) 00011000 (second data
reverse) 0 (p) 11 (stop) 111111
```

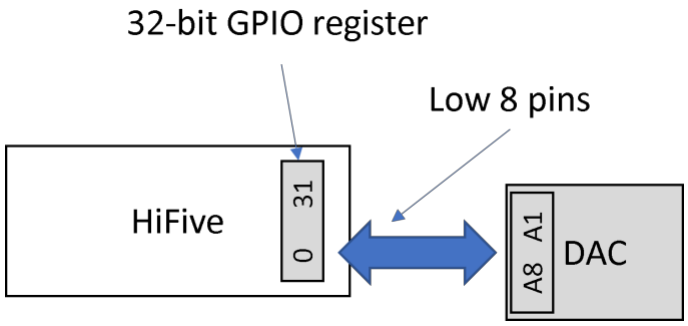
First data = reverse of (0b11010111) = 0b11101011

Parity is 1 but there are even number of "1"s in the data => error

Second data = reverse of (0b00011000) = 0b00011000

Parity is 0 and there are even number of "1"s in the data => no error

3. Complete the following code to set the output voltage to 10/256 volts. Assume that the DAC is a +10V output DAC.




```
void set_dac()

{

   uint32_t val = *(volatile uint32_t *) (GPIO_CTRL_ADDR +
GPIO_OUTPUT_VAL);

      // YOUR CODE HERE

   *(volatile uint32_t *) (GPIO_CTRL_ADDR + GPIO_OUTPUT_VAL) = val;

}
```

Vo = 10 * (A1/ 2 + A2/4 + A3/8 + A4/16 + A5/32 + A6/64 + A7/128 + A8/256)

= 40/256

When you solve the equation you get A6 = 1 and all other As == 0.

CODE:

val = val & 0xFFFFFF00;

val = val | 0x00000004;

4. Assume that you have a processor that is connected over a network adapter to the Internet. The network adapter receives packets with the following inter-arrival time distribution:

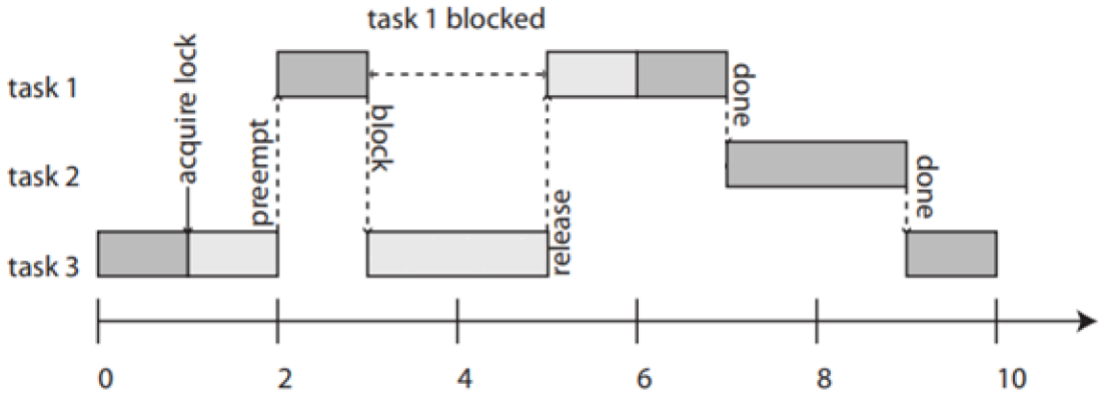
One of your EECS388 classmates developed a protocol called "adaptive interrupts" for notifying the microcontroller of packet arrivals at the network adapter:

By default, the network adapter is interrupt-driven, meaning that once a packet is received, it sends an interrupt to the processor immediately. After the first Interrupt is received, we switch to polling, meaning that the processor starts polling the network adapter for a short interval and then switch back to interrupts again.

Do you endorse your classmate's protocol? Why?

Yes! Polling work best when we have a burst of events, while interrupt saves CPU cycles when we have low activity. The adaptive interrupt protocol outlined above enjoys the best of both Polling and Interrupt.

5. What is the priority of task 3 at time 1, 4, and 6 with and without priority inheritance protocol? Assume that the initial priority of the tasks is: (highest priority) task#1 == 3 >task#2 == 2 > task#3 == 1 (lowerst priority)



time	1	4	6
Without priority inheritance	1	1	1
With priority inheritance	1	3	1

6. Consider the following real-time tasks and a single-core system.

Task	Compute Time (Seconds)	Period (seconds)
t1	3	10
t2	6	15
t3	10	35

L&L Bound

UB(1) = 1.0
UB(2) = 0.828
UB(3) = 0.779
UB(n) = 0.693

1.1. Which task has the highest priority?

Under RMS: t1

Under EDF the priority changes over time.

1.2. What is the total CPU utilization of the taskset?

0.9857

1.3. Is this taskset schedulable based on Liu & Layland bound?

We don't know!

1.4. Is this taskset schedulable under the rate monotonic scheduler? Use the exact analysis for your answer. The equation is given as follows where r denotes the response time (ready to completion) of the task. You do not need to upload your steps.

$$r_i^{k+1} = c_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i^k}{p_j} \right\rceil c_j, \quad \text{where } r_i^0 = \sum_{j=1}^{i} c_j$$

(A) Yes it is schedulable because $r_3^3 == r_3^2$ and $r_3^3 < 35$
(B) Yes it is schedulable because $r_3^4 == r_3^3$ and $r_3^4 < 35$
(C) No it is not schedulable because $r_3^3 > 35$
(D) No it is not schedulable because $r_3^2 > 35$

7. Draw a timeline (from time 0 to 40) of the taskset under the EDF scheduling on a piece of paper and answer the following questions.

7.1. Which task is running at time 9 and for how long?

(A) t1 is running until time 13
(B) t2 is running until time 10
(C) t3 is running until time 10

7.2. Which task is running at time 20 and for how long?

(A) t1 is running until time 23
(B) t2 is running until time 21
(C) t1 is running until time 21
(D) Both (A) and (B) are possible

7.3. Select all the correct statements?

(A) There is no missing deadline until time 40 using EDF scheduler

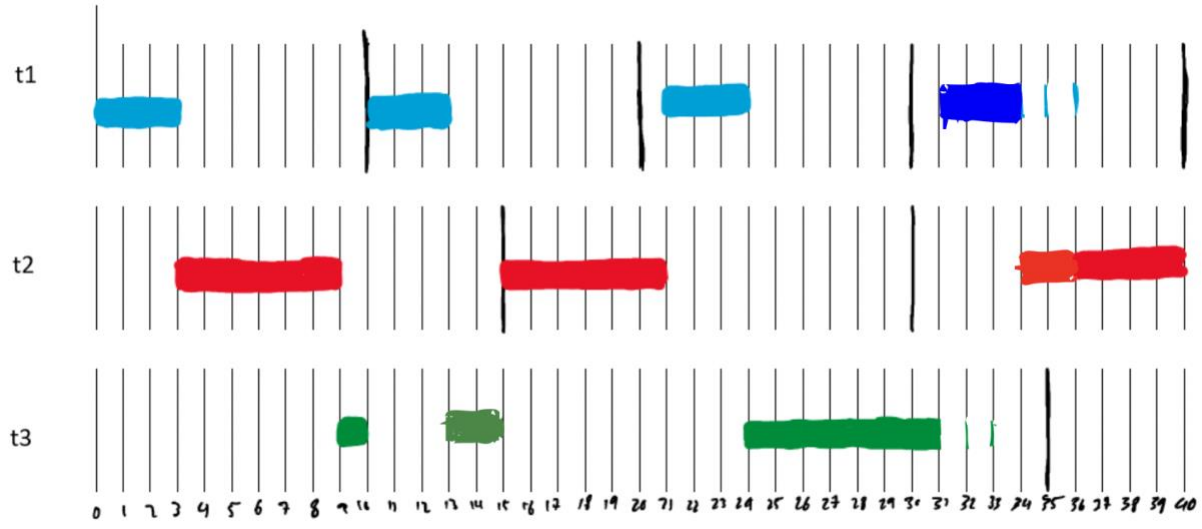
(B) t3 only runs for 1 seconds until time 20

(C) t1 is running at time 15

(D) t3 is running at time 24

(E) t3 is running at time 30

(D) None of the above



EECS388 Embedded Systems Fall 2022

HW 1 – Solution

Part 1 – True/False (1 point each)

Indicate true or false for each of the following statements:

1. Assembly code is generated before the machine code binary in a compiler toolchain. (T/F)

Ans: True.

2. You can manually perform everything that a makefile does. (T/F)

Ans: True.

3. Makefiles are bad for maintainability and portability compared to IDEs. (T/F)

Ans: False.

4. `Volatile` is a C type qualifier that provides specific instructions to the compiler on how the variables should be managed. (T/F)

Ans: True.

5. In 2's complement binary representation, the most significant bit is the sign bit. (T/F)

Ans: True

6. Each general-purpose computer has its own customized ISA. (T/F)

Ans: False.

7. Operands in an instruction hold input/output data. (T/F)

Ans: True.

8. A memory with higher addressability has a higher capacity. (T/F)

Ans: False.

9. General-purpose registers hold opcode of an instruction. (T/F)

Ans: False.

10. Components of a Von Neumann computer are Input/Output, Memory, Processing Unit, and Controller. (T/F)

Ans: True.

11. A Von Neumann computer is a little-endian computer. (T/F)

Ans: False.

12. In a Von Neumann computer model,

    a. IR and PC are part of the control unit. (T/F)
       Ans: True

    b. the next value of PC depends on the current value of IR. (T/F)
       Ans: True
       E.g.: JMP or BR change PC

    c. programs are stored in memory. (T/F)
       Ans: True

    d. the register file is part of the controller. (T/F)
       Ans: False

    e. controller decodes the instructions. (T/F)
       Ans: True

13. We have three types of instructions in LC-3: Operate, Date movement, and Control instructions. (T/F)

Ans: True.

14. We need 3 bits to address each register in a register file with 8 registers. (T/F)

Ans: True.

15. Every instruction in LC-3 needs to go through the FETCH phase of instruction processing. (T/F)

Ans: True.

16. Every instruction in LC-3 needs to go through the EVALUATE ADDRESS phase of instruction processing. (T/F)

Ans: False.
E.g.: Operate instructions.

17. In PC-relative addressing mode, the processor needs to perform one memory access to evaluate the final address. (T/F)

Ans: False.

18. Control instructions alter the sequential execution of instructions. (T/F)

Ans: True.
E.g.: JMP, BR, JSR...

Part 2 – Multiple Choice (2 points each)

Select the appropriate options for the following questions:

1. What is a cyber-physical system?
    a. A computer system that runs a specific application
    b. Another name for embedded systems
    c. A physical system that includes a computer inside
    d. All above

Ans: D

2. What are the differences between a general-purpose computer and an embedded system?
    a. Embedded systems only run one application, but general-purpose computers run many applications.
    b. Embedded systems are not programmable, but general-purpose computers can be programmed.
    c. Embedded systems usually have runtime constraints but for general-purpose computers, faster execution is always better.
    d. All above.

Ans: D

3. Which one is correct regarding the efficiency of embedded systems?
    a. Since embedded systems usually have power constraints, then it is important to reduce their power consumption.
    b. Embedded systems need to be optimized for size, power, weight, and cost.
    c. Performance and efficiency are equally important for embedded systems.
    d. All above.

Ans: D

4. Which one of the following variables can represent the decimal range of $[2^{32} -1, -2^{32}]$? (Assuming `int` type is 32 bits)
    a. int var;
    b. long int var;
    c. long long int var;

d. unsigned int var;

Ans: both B and C

If it was [$2^{31}-1$, -2^{31}] then, the answer also include "int var".

5. What does code (1) and (2) print after executing the main function?
   a. 10, 11
   b. 10, 10
   c. 11, 11
   d. 11, 10

```
// Code 1
void add(int a) {
    a = a + 1;
}

void main()
{
   int var = 10;
   add(var);
   printf("%d\n", var);
}
```

```
// Code 2
void add(int *a) {
    *a = *a + 1;
}

void main()
{
   int var = 10;
   add(&var);
   printf("%d\n", var);
}
```

Ans: A Code 1 prints 10 and Code 2 prints 11.

6. What does the following code print?

   a. 10, <Address of `var`>
   b. <Address of `var`>, 10
   c. 10, 10
   d. The code has errors

```
void main()
{
   int *var;
   …
   *var = 10;
   printf("%d, %d\n", var, *var);
}
```

Ans: If you execute the code in its current format you get a segmentation fault (Option D). But since we did not discuss memory

allocation in class, you get points if you also select option B (assuming that in the ellipsis there is a malloc statement).

7. When can we expect that the PC value does not change in LC-3?
    a. After executing an Operate instruction
    b. After executing a Data Movement instruction
    c. After executing a Control instruction
    d. None of the above

Ans: C

E.g., in the following code when we execute the JMP instruction the PC does not change:

```
        .ORIG  0x3000
LABEL   JMP    LABEL
```

8. Consider a Load instruction with PC-relative addressing mode. What is the range of addresses that the instruction can load from if the width of PC offset is 5 bits? Note that PC offset is a 2's complement number.
    a. [PC + 15, PC - 16]
    b. [PC + 16, PC - 15]
    c. [PC + 32, PC - 31]
    d. [PC + 31, PC - 32]

Ans: A

With 5 bits we can address $[-2^4, 2^4 - 1]$ -> [15, -16]

In general, with an n-bit 2's complement number, you can address the $[-2^{n-1}, 2^{n-1} - 1]$ range.

Part 3 – Short Answer (2 points each)

Provide a short answer to the following questions:

9. Why is a native compilation not usually an option for embedded systems?

Ans: Because of the limited resources.

10. Why do we need a makefile? Provide three reasons.

Ans: 1) Reduce Error.
2) Improve Scalability.
3) Its faster than a manual build.

11. Write two reasons for using a version control software such as git.

Ans: 1) Facilitate Collaboration.
2) Track Changes.

12. Why do we learn C in EECS388?

Ans: C is the dominant programming language in Embedded systems.

13. Write a C statement that allocates 8 bits in memory. (Single line of code)

Ans: char var;

14. What is the result of the following bitwise operations?

(signed) 10011011 << 2 =
(unsigned) 10011011 << 2 =

(signed) 10011011 >> 2 =
(unsigned) 10011011 >> 2 =
~ (signed) 10011011 =

Ans: 011 011 00
011 011 00
111 001 10
001 001 10
011 001 00

15. In C, a function cannot return multiple variables. What is a solution for modifying several input variables inside a C function?

Ans: Passing by pointer.

16. What are the two components of an LC-3 instruction? What info each of these components contains?

Ans: 1) Operand -> data to be used by the instruction.
2) Opcode -> operation to be performed.

17. The following is the machine code for an LC-3 instruction:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1

Answer the following question based on the LC-3 instruction format table.

a. Opcode is ……
b. Mnemonics is ……
c. (True/False) This instruction updates condition codes after execution. (T/F)
d. (Multiple choice) Bits [2:0] of the instruction represents
    i. Source Register
    ii. Destination Register

iii. Immediate value
iv. Opcode

e. (True/False) This instruction updates the register file after execution.

Ans: a) 0b 0001
b) ADD
c) True
d) Source register bit [5] ==0
e) True

18. What is the purpose of condition codes in LC-3?

Ans: It enables conditional branches.

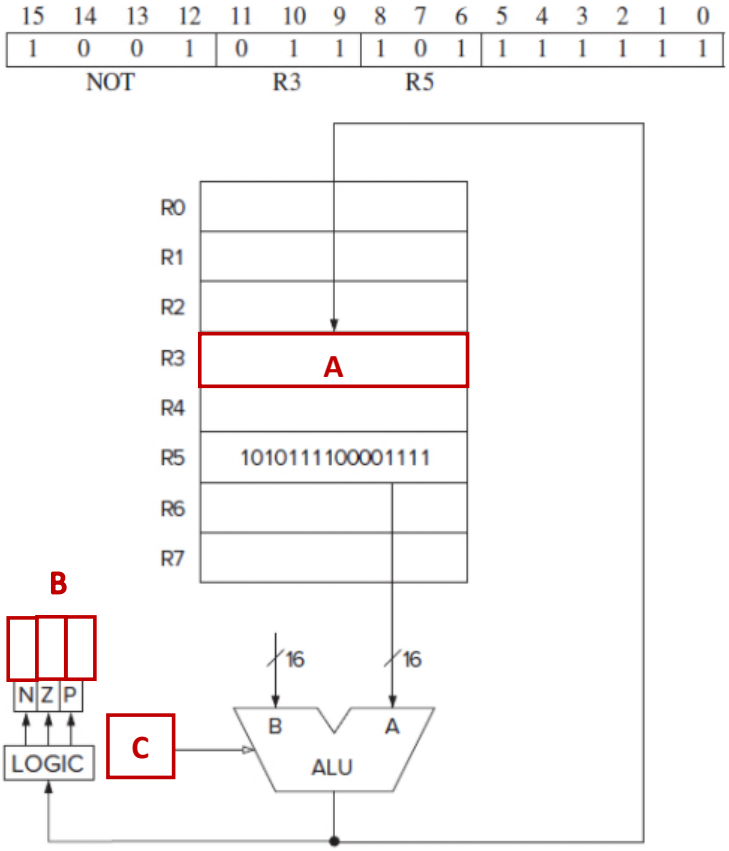
19. What is the value of the condition code registers (N, Z, P) after executing the following piece of code in LC-3?

```
AND   R1,R1, 0x00
NOT   R2,R1
```

Ans: 0x0000 => The result of NOT is 0xFFFF

0xFFFF => N = 1, Z = 0, P = 0

20. Assuming the following instruction is being executed in LC-3, fill in the blanks labeled as “A”, “B”, “C”.



Ans: A) 0101000011110000

B) 001

C) NOT

21. What is the purpose of "TRAP" instruction?

Ans: It requests service from the OS.

Part 4 – Other Questions (3 points each)

1. Write the LC-3 assembly representation of the following C code. Assume R1, R2, R3, R4, R5, R6 are in R1, R2, R3, R4, R5, R6 registers, respectively.

Ans:

```
if (R5 == R6)
      R1 = R2 + R3;
else
      R1 = R3 + R3;
R4 = R1 + R1;
```

What we need to do is to first use a sequence of Operate instructions to set the condition codes to a value which we can use to implement R5 == R6 condition. The easiest way is to subtract R5 from R6, if the result it zero (i.e., condition code is [Z=1, N=0, P=0]), then the condition of the if statement is true, otherwise it is false. Since we don't have SUB instruction in LC3, we need to use 3 instructions to implement SUB (refer to Slide#14-Lecture-5). The following is one way to implement the above C-code in LC3:

	NOT	R7, R6
	ADD	R7, R7, #1
	ADD	R7, R7, R5
	BRz	IF
	ADD	R1, R3, R3
	JMP	FINISH
IF	ADD	R1, R2, R3
FINISH	ADD	R4, R1, R1

2. Assume a 16-bit instruction with the following format:

OPCODE	DR	SR	IMM
4 Bits	5 Bits	5 Bits	

If there are 12 opcodes and 32 registers, what is the maximum number of bits that we can assign to the immediate field (IMM):

Ans:

- We need 4 bits to represent 12 opcodes
- We need 5 bits to address 32 registers

=> 16 – 4 – 5 – 5 = 2 bits remains for IMM

3. State different phases of executing an instruction. Briefly explain what operations take place in each phase.

Ans: 1) Fetch: Fetch the next instruction pointed by PC from the memory and increment PC

2) Decode: Decode the instruction type and generate appropriate control signals

3) Evaluate Address: Generate the memory address for accessing the memory

4) Fetch Operands: Read operands from memory or reg file

5) Execute: Execute the operation in ALU

6) Write results: Write results in the reg file or memory

4. Consider the following memory organization

Addr	Data (1 byte)
0x0	
0x1	
0x2	
0x3	
0x4	
0x5	
0x6	
0x7	

a. What is the addressability and address space of this memory?

Ans: Addressability = 1 byte

Address space = 8 unique addresses

b. Assuming that the memory is Little Endian and is initialized with 0s, what is the final content of the memory after storing the following multi-byte variable in the memory:

```
int short var = 0x5678;
assume &var -> 0x0;
```

Ans:

0x78
0x56
0x00
0x00
. . .
0x00

5. Fill in the table:

Decimal	2's complement?	Binary	Hexadecimal
0	NO	0b0	0x0
2	Yes	0b010	0x2
9	Yes	0b01001	0x09
-128	Yes	0b1000 0000	0x80
131	No	0b1000 0011	0x83
-32768	Yes	0b1000 0000 0000 0000	0x8000

.ORIG

- Where to place LC-3 program in memory

```
Memory Address
                01  ;
                02  ; Program to multiply an integer by the constant 6.
                03  ; Before execution, an integer must be stored in NUMBER.
                04  ;
                05          .ORIG   x3050
0x3050          06          LD      R1,SIX
0x3051          07          LD      R2,NUMBER
0x3052          08          AND     R3,R3,#0        ; Clear R3. It will
                09                                  ; contain the product.
                0A  ; The inner loop
                0B  ;
0x3053          0C  AGAIN   ADD     R3,R3,R2
0x3054          0D          ADD     R1,R1,#-1       ; R1 keeps track of
0x3055          0E          BRp     AGAIN           ; the iterations
                0F  ;
0x3056          10          HALT
                11  ;
0x3057          12  NUMBER  .BLKW   1
0x3058          13  SIX     .FILL   x0006
                14  ;
                15          .END
```

.FILL

- Initialize a memory location to a number or label

```
Memory Address
                01  ;
                02  ; Program to multiply an integer by the constant 6.
                03  ; Before execution, an integer must be stored in NUMBER.
                04  ;
                05          .ORIG   x3050
0x3050          06          LD      R1,SIX
0x3051          07          LD      R2,NUMBER
0x3052          08          AND     R3,R3,#0        ; Clear R3. It will
                09                                  ; contain the product.
                0A  ; The inner loop
                0B  ;
0x3053          0C  AGAIN   ADD     R3,R3,R2
0x3054          0D          ADD     R1,R1,#-1       ; R1 keeps track of
0x3055          0E          BRp     AGAIN           ; the iterations
                0F  ;
0x3056          10          HALT
                11  ;
0x3057          12  NUMBER  .BLKW   1
0x3058          13  SIX     .FILL   x0006
                14  ;
                15          .END
```

.BLKW

- Set aside a block of memory

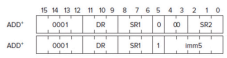
```
Memory Address
                01  ;
                02  ; Program to multiply an integer by the constant 6.
                03  ; Before execution, an integer must be stored in NUMBER.
                04  ;
                05          .ORIG   x3050
0x3050          06          LD      R1,SIX
0x3051          07          LD      R2,NUMBER
0x3052          08          AND     R3,R3,#0        ; Clear R3. It will
                09                                  ; contain the product.
                0A  ; The inner loop
                0B  ;
0x3053          0C  AGAIN   ADD     R3,R3,R2
0x3054          0D          ADD     R1,R1,#-1       ; R1 keeps track of
0x3055          0E          BRp     AGAIN           ; the iterations
                0F  ;
0x3056          10          HALT
                11  ;
0x3057          12  NUMBER  .BLKW   1               ; Number of blocks
0x3058          13  SIX     .FILL   x0006
                14  ;
                15          .END
```

.END

- Tells assembler it has reached the end of program
- Just a delimiter marking the end of program for "Assembler" not processor
  - This is different from HALT!

ADD

```
ADD DR, SR1, SR2
ADD DR, SR1, imm5
```



- Operation:

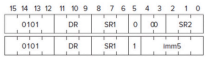
```
If (bit[5] == 0)
     DR = SR1 + SR2
Else
     DR = SR1 + SEXT(imm5)
setCC();
```

- Example:

```
ADD R1, R2, R3
ADD R2, R5, #7
```

AND

```
AND DR, SR1, SR2
AND DR, SR1, imm5
```



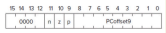
- Operation:

```
If (bit[5] == 0)
     DR = SR1 & SR2
Else
     DR = SR1 & SEXT(imm5)
setCC();
```

- Example:

```
AND R1, R2, R3
AND R2, R5, #7
```

BR



```
BRn LABEL     BRz LABEL     BRp LABEL
BRnp LABEL    BRzp LABEL    BRnz LABEL
BRnzp LABEL
```

- Operation:

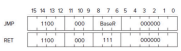
```
If ((n AND N)OR (p AND P) OR (z AND Z))
     PC = PC + SEXT(PCoffset9)
```

- Example:

```
BRzp   LOOP
BRnzp TARGET
```

JMP (Jump), RET (Return from Subroutine)

```
JMP BaseR
RET
```



Operation:

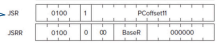
```
JMP: PC = BaseR
RET: PC = R7
```

- Example:

```
JMP R1
RET
```

JSR, JSRR (Jump to Subroutine)

```
JSR LABEL
JSRR BaseR
```



- Operation:

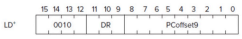
```
Temp = PC;
if (bit[11] == 0)
     PC = BaseR
else
     PC = PC + SEXT(PCoffset11)
R7 = TEMP
```

- Example:

```
JSR QUEUE
JSRR R3
```

LD (Load)

```
LD   DR, LABEL
```



- Operation:

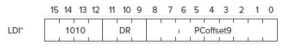
```
DR = Mem[PC + SEXT(PCoffset9)]
setCC();
```

- Example:

```
LD R4, VALUE
```

LDI (Load Indirect)

```
LDI  DR, LABEL
```



- Operation:

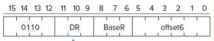
```
DR = Mem[Mem[PC + SEXT(PCoffset9)]]
setCC();
```

- Example:

```
LDI R4, InDirADDRESS
```

LDR (Load Base+Offset)

```
LDR DR,BaseR,Offset6
```



- Operation:

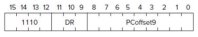
```
DR = Mem[BaseR + SEXT(offset6)]
setCC();
```

- Example:

```
LDR R4,R2,#-5
```

LEA (Load Effective Address)

```
LEA  DR,LABEL
```



- Operation:

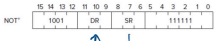
```
DR = PC + SEXT(PCoffset9)
setCC();
```

- Example:

```
LEA R4,TARGET
```

NOT (Bitwise complement)

```
NOT  DR,SR
```



- Operation:

```
DR = NOT(SR)
setCC();
```

- Example:

```
NOT R4,R2
```

ST (Store)

```
ST   SR, LABEL
```

- Operation:

```
Mem[PC + SEXT(PCoffset9)] = SR
```

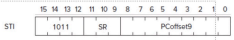
- Example:

```
ST R4, HERE
```

LDI

STI (Store Indirect)

```
STI  SR, LABEL
```



- Operation:

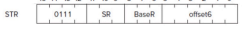
```
Mem[Mem[PC + SEXT(PCoffset9)]] = SR
```

- Example:

```
STI R4, InDirADDRESS
```

STR (Store Base+Offset)

```
STR  SR,BaseR,Offset6
```



- Operation:

```
Mem[BaseR + SEXT(offset6)]= SR
```

- Example:

```
STR R4,R2,#-5
```

TRAP (System Call)

```
TRAP  trapvect8
```

- Operation:

```
R7 = PC
PC = mem[ZEXT(trapvect8)]
```

- Example:

```
TRAP   x23
```

*memory location 0x0000 – 0x00FF implement Trap Vector Table

EECS388 Embedded Systems
Midterm Exam – Fall 2022

Name:

KU ID:

Key

Exam Guidelines:

1. You have **75 minutes** to complete this exam.
2. This exam is open-book and open-note.
3. Show your work and write your assumption if you wish to receive partial credit.
   Note: You do not get any points if there is a mismatch between your notes and your final answer.
4. The exam is meant to test your understanding. So be patient and read the questions/problems carefully before you answer.
5. The exam comes with a cheat sheet that includes necessary information for answering questions without accessing the Internet or course materials.
6. A percentage of the exam is extra credit (depending on the performance of the class).

DO NOT do anything that might be perceived as cheating.

- **We have zero tolerance for cheating.**
- There are several implicit mechanisms in place for preventing you from cheating and possibly detecting students who cheat. E.g., the questions and options are reordered, and there are different versions of the same exam.

Exam has 3 parts: part 1: 10 questions; part 2: 7 questions; part 3: 7 questions

Part 1 – True/False (1 point each – total 10 points)

Indicate true or false for each of the following statements.

1. An embedded system is either optimized for power or performance (T/F)

2. `short` and `long` are type modifiers in C that decrease and increase the size of data types. (T/F)

3. x86 and ARM are two examples of standard ISAs that are widely used in many processors. (T/F)

4. A memory that has higher *addressability* has higher capacity (i.e., stores more bits). (T/F)

5. Components of a Von Neumann computer are Input/Output, Memory, Processing Unit, and Controller. (T/F)

6. Endianness (i.e., little endian or big endian) defines how a processor stores a multi-byte variable in memory. (T/F)

7. A Von Neumann computer stores both data and instructions in the memory. (T/F)

8. Control instructions change the value of PC. (T/F)

9. In the FETCH phase of instruction processing, an instruction is read from memory, and PC is incremented. (T/F)

10. In STI instruction, we have one memory-read (to get the address), and one memory-write to update the memory. (T/F)

Part 2 – Multiple Choice (Total 15 points)

Select the appropriate option(s) for the following questions:

1. The 2's complement hex number 0xFA19 *[multi option – 3.5 points (+0.5 for correct selection, +0.5 for not selecting incorrect options)]*
   a. is a positive number
   b. is a negative number
   c. has "1" as its sign bit
   d. has "1" as its least significant bit
   e. has "0" as its most significant bit
   f. is smaller than the 2's complement hex number 0xFF
   g. is a 16-bit binary number

2. What is the minimum number of bits that we need to address each register in a register file with 33 registers? *[single option – 2 points]*
   a. 4
   b. 5
   c. 6
   d. 7
   e. 8

3. We have a memory with the following characteristics:
   Addressibility = 32 bits
   Address space = 48 memory blocks

   We need at least ….. number of bits to address each memory block. *[single option – 2 points]*
   a. 3
   b. 4
   c. 5
   d. 6
   e. 7

4. Which one of the following variables can represent the decimal range of $[2^{26}, -2^{31}]$? (Assuming `int` type is 32 bits) *[multi option – 2.5 points (+0.5 for correct selection, +0.5 for not selecting incorrect options)]*
   a. short var;
   b. int var;
   c. long int var;
   d. signed int var;
   e. unsigned int var;

5. What is the type of "ptr" in the following code? *[single option – 1 point]*
   a. Integer

b. Floating point
c. Enumerated
(d.) Derived (pointer)
e. Void

```
void main()
{
   int var = 10;
   int *ptr;
   ptr = &var;
   printf("%d,%d\n", &var, ptr);
}
```

6. Consider a Load instruction with PC-relative addressing mode. What is the range of addresses that the instruction can load from if the width of PC offset is 3 bits and PC offset is an **unsigned number**? *[single option – 2 points]*
   a. [PC + 15, PC - 16]
   b. [PC + 15, PC]
   c. [PC + 7, PC - 8]
   d. [PC + 8, PC]
   e. [PC + 4, PC]
   f. [PC + 3, PC]
   g. [PC + 3, PC - 4]

[PC+8, PC+1]

+2

No correct option

7. We often perform cross-compilation for embedded system applications because *[multi option – 2 points (+0.5 for correct selection, +0.5 for not selecting incorrect options)]*
   (a.) embedded systems often have limited resources to compile the application
   b. embedded systems do not have a processor
   (c.) native compilation on the embedded system is often too slow
   d. native compilation only runs on a Windows operating system

Part 3 – Short Answer (Total 20 points)

Provide a short answer to the following questions:

1. What is the value of var (in hexadecimal) after executing the following C code? [2 *points*]

```
short int var = 0x2102;
var = var & 0x82F1;
var = var ^ 0x1111;
```

0x0000

0b 0010 0001 0000 0010 &
0b 1000 0010 1111 0001
0b 0000 0000 0000 0000 (+1)

var = 0x1111 (+1)

partial credit

2. What is the result of the following bitwise operations in hex? *[0.75 points each – total 3 points]*

0b 1111 0111 << 2 => 0b 1101 1100 = 0xDC

(+0.75)

(signed) 0xF7 << 2 = 0x DC
(signed) 0x86 >> 2 = 0x E1
(unsigned) 0x91 >> 2 = 0x 24
NOT 0x11 = 0x EE

0b 1000 0110
0b 0010 0100

NOT 0b 0001 0001
0b 1110 1110

3. Fill in the memory content after executing the following LC-3 assembly code. Assume that the initial value of all memory locations is 0x0000. Note: The end of the string is a NULL character with ASCII code 0x00. *[3 points]*

```
.ORIG          x3002
.STRINGZ "388"
```

Address	Data (2 bytes)
0x3000	0x0000
0x3001	0x0000
→0x3002	0x0000 0x0033 ←
0x3003	0x0000 0x0038 ←
0x3004	0x0000 0x0038
0x3005	0x0000 0x0000
0x3006	0x0000

(+0.75)

(+0.75 each ½)

(+0.25 for NULL char)

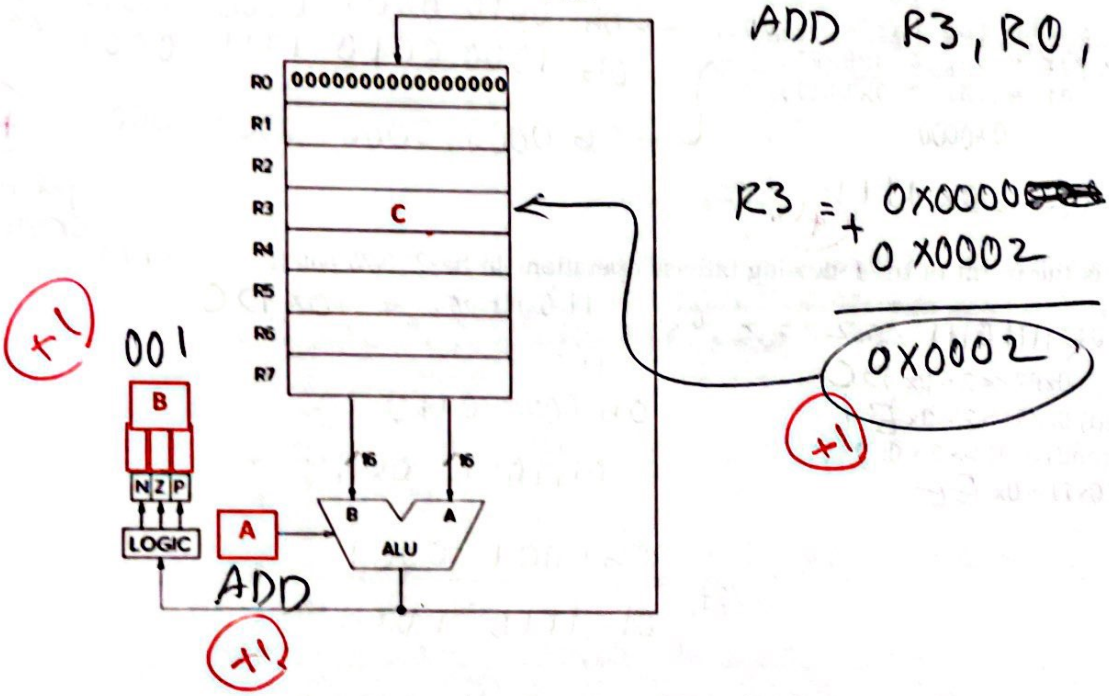
4. Assuming the following instruction is being executed in LC-3, fill in the blanks labeled as "A", "B", "C". *[3 points]*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	1	0	0	0	1	0	0	0	1	0

Imm

R3 R0

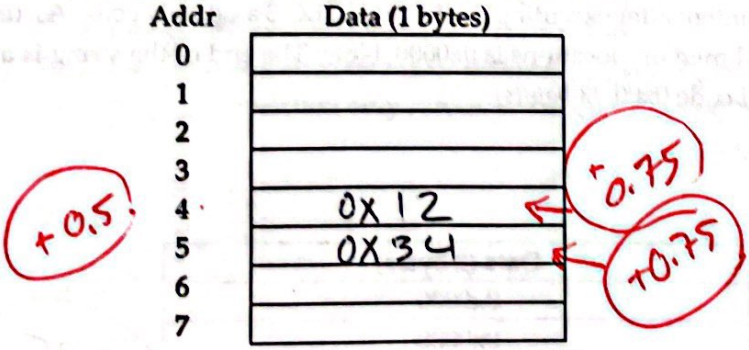
ADD R3, R0, 05 00010



R3 = + 0X0000 / 0X0002 = 0X0002

+1 +1 +1

5. Consider the following memory organization.



+0.5 +0.75 +0.75

a) What is the addressability and address space of this memory? *[1 point]*

1Bytes 8

+0.5 +0.5

b) Assuming that the memory is Big Endian. What would be the memory content after storing the following multi-byte variable in the memory? Assume that memory is initialized to zero. *[2 points]*

```
Unsigned short int var = 0x1234;
assume &var -> 0x4;
```

6. What is the value of R1 after executing the following LC-3 assembly with the following memory content? *[2 points]*

```
        .ORIG x3000
        LDI     R1, ADDRESS
ADDRESS .BLKW 1
```

0X 3005

Address	Data (2 bytes)
0x3000	0xA200
0x3001	0x3003
0x3002	0x3004
0x3003	0x3005
0x3004	0x3006
0x3005	0x3007
0x3006	0x3001

7. Write the LC-3 assembly representation of the following C code. Assume R0, R1, R2, R3, R4 are in R0, R1, R2, R3, R4 registers, respectively. *[4 points]*

```
if (R4)
    R0 = R1 & R2;

R3 = R3 + R1;
```

```
        ADD   R4, R4, #0
        BRnp  LABEL
        JMP   EXIT
LABEL   AND   R0, R1, R2
EXIT    ADD   R3, R3, R1
```

+1 setting CC

+1 correct branching

+1 correct operands/opcode/format

+1 correct functionality

EECS388 Embedded Systems: Sample Question

Question 1: How much should you increment the PC after fetch stage in a processor that is a word addressable (1 word=32 bit) and the width of each instruction is 4 Bytes? (4 points)

Answer:

a. 1 (__)
b. 2 (__)
c. 8 (__)
d. 4 (__)

Question 2: Let's assume that a 32 bit variable with value of AABBCCDD in hexadecimal has to be stored in the memory. Assume that each memory address can store **8 bits** and the processor uses **Little Endian** method to store data. Write down the hex digits to be stored in different addresses under data column of the table below. (4 points)

Answer:

Address	data
0x00	DD
0x01	CC
0x02	BB
0x03	AA

Question 3: What are the main components of a single DRAM cell? (4 points)

Answer:

a. Transistor (__)
b. flip-flops (__)
c. inverters (__)
d. capacitor (__)

Question 4: What are the advantages of on-chip SRAM memory over DRAM? (4 points)

Answer:

a. Read/write speed (latency) (__)
b. Ease of fabrication with the processor (__)
c. Cost per bit (__)
d. non-volatile (__)

Question 5: Consider the following C code. Let us assume that the function sum() is currently executing. What is the address stored in the return address register 'ra'? (4 points)

```
int sum(int a, int b){
  return a+b;
}
void main() {
  int x=20;
  int y=10;
  int z= sum(x,y);
  z++;
  y=z+1;

}
```

Answer:

a. Memory address that stores line "z++;" (__)
b. Memory address that stores line "y=z+1;" (__)
c. Memory address that stores line "return a+b;" (__)
d. Memory address that stores line "int z= sum(x,y);" (__)

Question 6: Why embedded systems primarily use the C programming language? (4 points)

Answer:

a. Supports bitwise operations (__)
b. C language is architecture-dependent (__)
c. Secure against buffer overflow attack (__)
d. Lower code overhead for execution (__)

Question 7: For the following assembly program, how many times the Addi instruction inside the loop will execute? All instructions are based on 32 bit MIPS ISA. (4 points)

```
addi r1, r0, 4
Loop:
addi r1, r1, -1
bne r1, r0, Loop
sub r1, r0, r0
```

Answer:

a. 0 (__)
b. 1 (__)
c. 4 (__)
d. 5 (__)

Question 8: What should the MIPS assembly for the following C code?

```
If (a<b) {
 goto Label1; }
```

Assumptions: a is in $r1, b is in $r2. Use two instructions.

Answer:

```
slt   $r3, $r1, r2          # if $r1<$r2, make #r3=1
bne $r3, $r0 , Label1       # goto Label1, if $r3 not equal to 0
```

Question 9: Mark the cases inside the brackets for which you cannot execute the same assembly program compiled for one microprocessor in another one. (4 points)

Answer:

a. If both microprocessors have the same ISA (___)
b. If both microprocessors have the different ISA and different microarchitecture (___)
c. If both microprocessors have the same ISA but different microarchitecture (___)
d. If both microprocessors have different ISA (___)

Question 10: Mark the multiplications that can be performed using logical left shift? Assume Z=00000111 in binary. (4 points)

Answer:

a. Z x 9 (___)
b. Z x 16 (___)
c. Z x 32 (___)
d. Z x 64 (___)

Question 11: Which of the following statements (if any) are true for the code used in the lab below. (4 points)

```
1  #include <stdint.h>
2  #include "eecs388_lib.h"
3  int main()
4  {
5      int gpio = GREEN_LED;
6
7      gpio_mode(gpio, OUTPUT);
8
9      while(1)
10     {
11         gpio_write(gpio, ON);
12         delay(1000);
13         gpio_write(gpio, OFF);
14         delay(300);
15     }
16 }
```

a. The while(1){…} loop iterates just once (___)
b. The gpio_write() in line 11 is a function declaration (___)
c. We may not need such while loop if an operating system is present (___)
d. The eecs388_lib.h contains the definition of function gpio_write() (___)

Question 12: which of the following memory technologies are volatile? (4 points)
Answer:

a. DRAM (___)
b. Hard drive (___)
c. Flash memory (___)
d. Registers (___)

Question 13: Which of the following statements are true for port mapped and memory mapped I/O? (4 points)
Answer:

a. Port mapped I/O uses only load and store instructions to communicate with I/O devices (___)
b. In memory mapped I/O, the I/O memory is mapped into the CPU address space (___)
c. In port mapped I/O, the I/O memory is mapped into the CPU address space (___)
d. The Hifive platform used in your lab uses memory mapped I/O (___)

Question 14: For the C code presented below, indicate the section in memory layout each variable is stored or makes use of. Circle the correct option for each question. (BSS is Uninitialized Data Segment and Data is Initialized Data Segment). (8 points)

```
int globA=10;
int globB=0;
int main () {
   int varA;
   int varB = 10;
   static int varC = 1;
   static int varE = 0;
   char *varD;
   varD = (char*)malloc(8);
   varA = varB + varC;
   return varA;
}
```

Answer:

a. int globB=0; (Stack—Heap—*BSS*—Data)
b. int varA; (*Stack*—Heap—BSS—Data)
c. int varB = 10; (*Stack*—Heap—BSS—Data)
d. static int varC = 1; (Stack—Heap—BSS—*Data*)
e. char *varD; (*Stack*—Heap—BSS—Data)
f. varD = (char*)malloc(8); (Stack—*Heap*—BSS—Data)

Question 15: What information is **not stored** in program counter register? (4 points)

Answer:

a. Current instruction (__)
b. Next instruction (__)
c. Address of currently executing inst (__)
d. Address of next instruction (__)

Question 16 For the following assembly program, what is the content in memory address 100 after executing the following code? All numbers are decimal. Please note that register r0 contains zero. All instructions are based on MIPS ISA.

```
addi r1,r0, 54
addi r2,r0, 58
addi r3,r0, 100
sw r1, 0(r3)
sw r2, 4(r3)
```

Answer:

a. 54 (__)
b. 58 (__)
c. 100 (__)
d. 4 (__)