

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, Data 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 $[-2^{32}, -1]$? (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

$[-2^4, 2^4 - 1] \rightarrow [15, -16]$

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) $10011011 \ll 2 =$

(unsigned) $10011011 \ll 2 =$

(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.

- Opcode is
- Mnemonics is
- (True/False) This instruction updates condition codes after execution. (T/F)
- (Multiple choice) Bits [2:0] of the instruction represents
 - Source Register
 - 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?

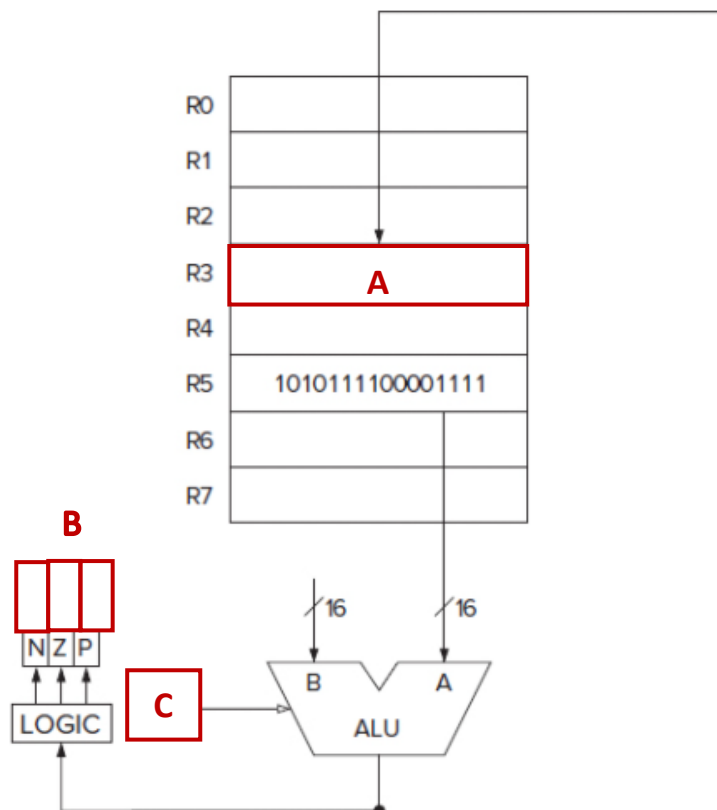
AND	R1, R1, 0x00
NOT	R2, R1

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".

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	1	1	1	0	1	1	1	1	1	1	1
NOT				R3			R5								



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 is 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

$$\Rightarrow 16 - 4 - 5 - 5 = 2 \text{ bits remains for IMM}$$

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