# 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)

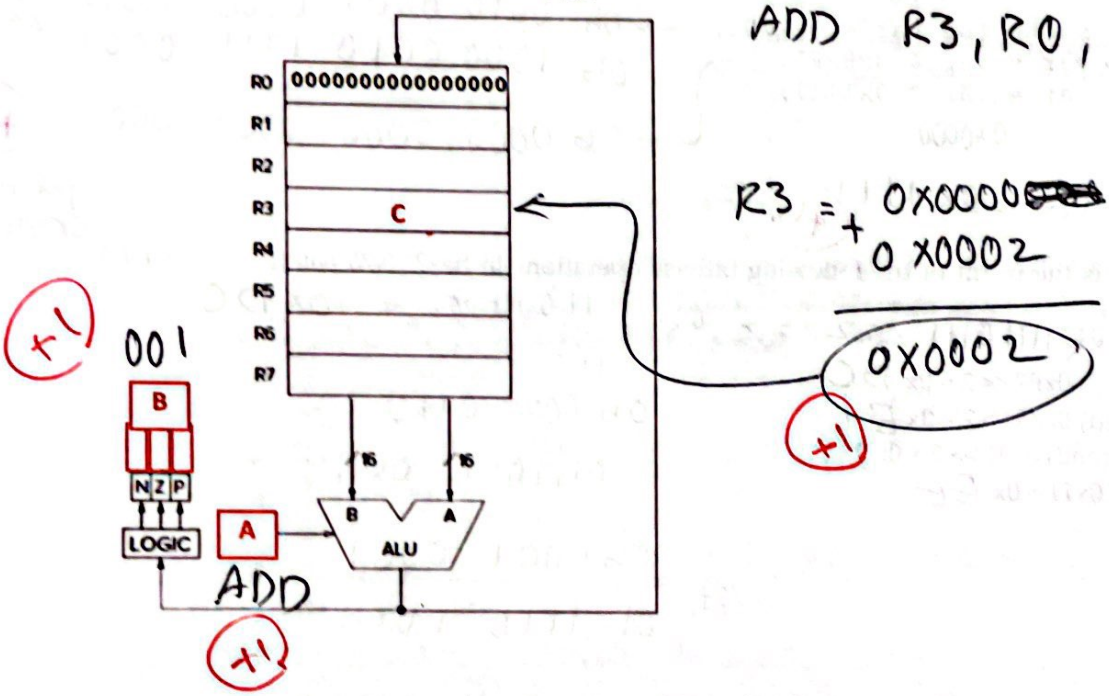
4. Assuming the following instruction is being executed in LC-3, fill in the blanks labeled as "A", "B", "C". *[3 points]*

| 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

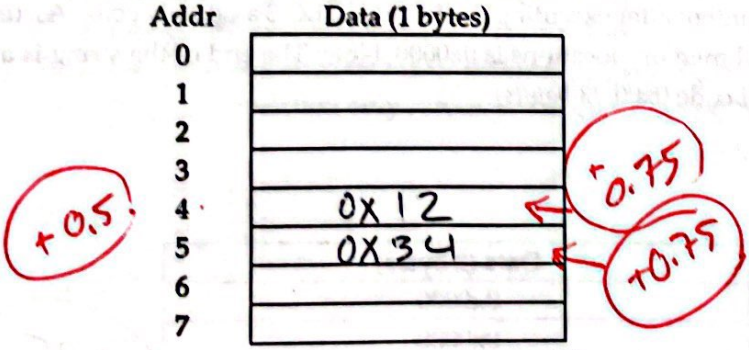
ADD R3, R0, 05 00010




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