

McGill University
Department of Electrical and Computer Engineering
ECSE 324 Mid-Term Exam Fall 2021

Name: _____

McGill ID: _____

Closed Book Exam. No calculator allowed. Answer all questions directly on the exam paper. You can do rough work on the blank pages. Rough work will not be used for grading.

Part 1 _____/10

Part 2 _____/07

Part 3 _____/08

Total _____/25

Part A: General knowledge

[10 points]

A.1: multiple choice questions [1 point] each.

Only select a single answer. If more than one answer is selected, you will get zero for that question.

1. Moore's Law states that:

- a. Single-thread performance doubles every two years
- b. The transistor count doubles every two years
- c. Clock frequency doubles every two years
- d. The number of logical cores doubles every two years
- e. A finite-state machine's outputs are only determined by its current state

2. What is -1 in 4-bits two's complement?

- a. 0001
- b. 1001
- c. 1111
- d. 1110
- e. None of these

3. Assuming a 32-bits word-aligned CPU. Which of the following effective addresses is valid when used by the LDR instruction to access memory:

- a. 0x00003001
- b. 0x00003002
- c. 0x00003003
- d. 0x00003004
- e. None of these

4. On an ARM machine, which of these instructions might modify the CPSR?

- a. BLE LOOP
- b. ADDGES R0, R1, R2
- c. LDR R0, [R1]
- d. MOVGT R1, #3
- e. None of these

5. Consider the following ARM assembly program, whose first instruction is at address 0x00000000 after assembling:

```
LOOP:    ADD R1, R1, #1
         CMP R1, R2
         BLE LOOP
```

What is the encoding of the branch target address?

- a. 0xFFFFFC
- b. 0xFFFFFE
- c. 0x000002
- d. 0x000004
- e. 0X000000

6. Who designed the first compiler, "A-0 System", in 1952?

- a. John Von Neumann
- b. Grace Hopper
- c. Steve Jobs
- d. Gordon Moore
- e. Ada Lovelace

7. What does it mean for an I/O device's register to be memory-mapped?

- a. The content of the register is stored in the main memory
- b. The content of the register can be accessed with load/store instructions
- c. The register must hold 32 bits of data
- d. The register is in the CPU
- e. None of the above

A.2 short answers [1point] each

8. Consider the following ARM instruction:

```
LDR, R2, [R6], #4.
```

Assuming R2 contains 0x00000002 and R6 contains 0x0000000A. After the instruction executes, what is the content of R6?

0x0000000E

9. Assume the following content is stored in a byte-addressable memory:

Address	Content
0x00000000	0x12
0x00000001	0x34
0x00000002	0x56
0x00000003	0x78

Assuming that you are dealing with a Little Endian ARM processor, what is the content of R0 after these two instructions have finished executing?

```
MOV R0, #0  
LDR R0, [R0]
```

0x78563412

10. Rewrite this sequence of ARM instructions using a single instruction while preserving the semantic
(tip: use the ARM Instruction Set Quick Reference CARD if you are not familiar with the instructions):

```
ADD R0, R1, R2  
CMN R1, R2
```

ADDS R0, R1, R2

B) ISA & Assembly

[7 points]

Assume a 24-bit RISC CPU with some general purpose registers (including PC): registers and instructions are both 24 bits wide, and the address size is 24 bits. The memory is byte-addressable and the only instructions available in the instruction set are:

- `ADD Rd, Rs1, Rs2 // Rd <- Rs1 + Rs2`
- `SUB Rd, Rs1, Rs2 // Rd <- Rs1 - Rs2`
- `MUL Rd, Rs1, Rs2 // Rd <- Rs1 * Rs2`
- `LD Rd, Rs1, Rs2 // Rd <- MEM[Rs1+Rs2]`
- `ST Rd, Rs1, Rs2 // MEM[Rs1+Rs2] <- Rd`
- `MOV Rd, #imm // Rd <- #imm (immediate value)`
- `BEQ Rs1, Rs2, displacement // PC <- PC+3*displacement if Rs1==Rs2`
- `BLT Rs1, Rs2, displacement // PC <- PC+3*displacement if Rs1<Rs2`

1. What is the minimal number of bits required to encode the opcode of an instruction? (i.e. identify the instruction). **[1 point]**

3

2. What is the maximum number of general purpose registers that this ISA could support? (*tip: think of the space available in the instruction to encode the register number*) **[1 point]**

128

3. What is the total memory addressable by this CPU in MB? **[1 point]**

16MB

4. Implement the following instruction using only the instructions from the ISA above. **[2 points]**

`BLE Rs1, Rs2, displacement // PC <- PC + 3*displacement if RS <= Rs2`

Assume displacement is a positive number.

(1pt for correct BEQ, BLT sequence, 1pt for correct displacement+1)

*BEQ Rs1, Rs2, displacement+1
BLT Rs1, Rs2, displacement*

The following is copied from the previous page for your convenience.

Assume a 24-bits RISC CPU with some general purpose registers (including PC): registers and instructions are both 24 bits wide, and the address size is 24 bits. The memory is byte-addressable and the only instructions available in the instruction set are:

- ADD Rd, Rs1, Rs2 // Rd \leftarrow Rs1 + Rs2
- SUB Rd, Rs1, Rs2 // Rd \leftarrow Rs1 - Rs2
- MUL Rd, Rs1, Rs2 // Rd \leftarrow Rs1 * Rs2
- LD Rd, Rs1, Rs2 // Rd \leftarrow MEM[Rs1+Rs2]
- ST Rd, Rs1, Rs2 // MEM[Rs1+Rs2] \leftarrow Rd
- MOV Rd, #imm // Rd \leftarrow #imm (immediate value)
- BEQ Rs1, Rs2, displacement // PC \leftarrow PC+3*displacement if Rs1==Rs2
- BLT Rs1, Rs2, displacement // PC \leftarrow PC+3*displacement if Rs1<Rs2

5. Implement the following instruction using only the instructions from the ISA above. **[2 points]**

```
BL displacement // LR  $\leftarrow$  PC of next instruction;  
                // PC  $\leftarrow$  PC + 3*displacement
```

Assume the PC is register R5 and the LR is register R6. Further assume that in this processor, the PC of the currently executing instruction is always two instructions ahead (like on ARM). Also assume displacement is a positive number.

1 point for updating LR register (-0.5pt if got PC+0 wrong)

1 points for updating PC correctly with BEQ instruction (or more complex scheme)

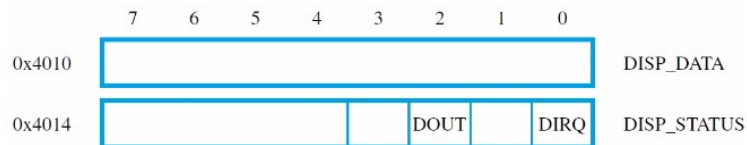
```
MOV R0, #0  
ADD R6, R5, R0 // LR  $\leftarrow$  PC+0
```

```
BEQ R0, R0, displacement // PC  $\leftarrow$  PC + 3* displacement
```

C) I/O Device & Function Call

[8 points]

Suppose that we wish to print a null terminated string (last byte of the string is 0) on a display. The display memory mapped I/O registers are shown below:



Fill in the blanks (underlined>) in the following assembly program to achieve this.

```
disp:          .word      0x00004010
str:           .ascii    "hello world\0"

_start:
              LDR        A1, =str
              BL         printString
end:          B

printString:
              PUSH      {V1}
              MOV       V1, A1

loop:
              LDRB     A1, [V1]
              CMP      A1, #0
              BEQ      return
              PUSH     {LR}
              BL       printChar
              POP      {LR}
              ADD     V1, V1, #1
              B        loop

return:
              POP      {V1}
              BX      LR

printChar:
              PUSH     {V1,V2}
              LDR     V1, disp

writeWait:
              LDRB     V2, [V1, #4]
              TST     V2, #4
              BEQ     writeWait
              STRB    A1, [V1]
              POP     {V1,V2}
              BX     LR
```