ECSE 324 Winter 2021 Midterm

Part A: Choose wisely

Part A.1: *Multiple select*. Choose as many options are as appropriate for each question.

1. Which of the following are 4-bit 2's complement numbers?
    a. 0b1010
    b. 0b0101
    c. 0b1111
    d. 0b0000

2. Under which of the following circumstances is there *overflow*?
    a. Unsigned addition: 0b1010 + 0b0111
    b. Unsigned subtraction: 0b10101 – 0b0111
    c. 2's complement addition: 0b1010 + 0b0111
    d. 2's complement subtraction: 0b1010 – 0b0111

3. Assume a 32-bit, word-aligned RISC processor. Which of the following addresses are appropriate values for PC?
    a. 0xCAFE
    b. 0xBEEF
    c. 0xACDC
    d. 0xACAB

4. Which of the following programs determine the target address of branch instructions?
    a. Compiler
    b. Assembler
    c. Linker
    d. Loader

5. Consider the following sequence of ARM instructions. Assuming they are fetched and decoded in order, which of them will execute? Assume R4 = 7.
    a. `CMP  R4, #8`
    b. `ADDLT R0, R0, #1`
    c. `MOVLE R1, #4`
    d. `SUBGT R2, R2, #4`

Part A.2: *Matching*. Select the appropriate option for each part of each question.

6. What are the defining characteristics of (i) polling, and (ii) interrupts?
    a. Processor repeatedly checks device status
    b. Hardware notifies processor when ready
    c. Prior to servicing a device, processor state is saved on the stack
    d. A special function is invoked to service the device

7. For each option, select among (i) true for `LDR R0, y`; (ii) true for `LDR R0, =y`; (iii) true for both; and (iv) true for neither.
    a. Uses PC-relative addressing.
    b. Uses SP-relative addressing.
    c. Puts a pointer to y in R0.
    d. Puts the value at y in R0.

Part A.3: *Short answer*.

8. Assume a 32-bit *big endian* RISC computer. If R0 = 0x0000 0004, what is in the *most significant byte* of R1 after: `Load R1, [R0]`?

9. Write a single ARM instruction that performs the same operations as the following instructions:

```
MOV R0, #8
MUL R2, R1, R0
STR R3, [R4, R2]
```

10. How many memory accesses are performed over the entire course of executing the instruction `PUSH {V1, V2, LR}`?

Addr	**Data**
0x00	0xD1
0x01	0x4B
0x02	0x45
0x03	0xC4
0x04	0x90
0x05	0x12
0x06	0x4F
0x07	0xEE
0x08	0x78
0x09	0x91
0x0A	0x03
0x0B	0x70
0x0C	0xB3
0x0D	0xDA
0x0E	0x7F
0x0F	0xE6

Part B: Retro RISC

Assume an 8-bit RISC CPU with four general purpose registers (including stack pointer SP), program counter PC, and current program status register CPSR.

The only available instructions are:

- `LD  Rd, Rbase  // Rd <- Mem[Rbase]`
- `ST  Rd, Rbase  // Mem[Rbase] <- Rd`
- `MOV  Rd, Rs  // Rd <- Rs`
- `MOV  Rd, #imm  // Rd <- #imm`
- `ADD  Rd, Rs  // Rd <- Rs + Rd`
- `CMP  Rd, Rs  // If (Rd – Rs == 0), set Z flag to 1`
- `B  displacement  // PC <- PC + displacement`

Each instruction can be conditionally executed (when Z==1), or always executed, based on a 1-bit condition field in each instruction.

What is the minimum number of bits required to encode a general purpose register operand?

- 1 bit
- 2 bits
- 3 bits
- 4 bits
- None of these

What is the maximum number of bits available to encode the immediate value for MOV?

- 1 bit
- 2 bits
- 3 bits
- 4 bits
- None of these

What is the largest displacement (in bytes) possible for the B instruction?

- 4 bytes
- 7 bytes
- 8 bytes
- 15 bytes
- None of these

What is the total memory addressable by this CPU?

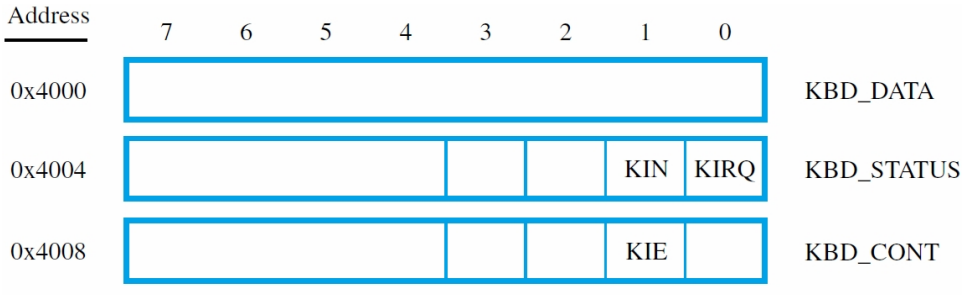
- 128 B
- 256 B
- 512 B
- 1 KB
- None of these

How many instructions are required to execute something equivalent to PUSH {R1, R2, R3} in this architecture? (Do not make any assumptions about any register contents besides SP, which points to the top of the stack.)

- 4 instructions
- 5 instructions
- 6 instructions
- 7 instructions
- None of these

Part C: Get it, got it, good

Complete the following assembly to implement a function that gets a string of characters from the keyboard using polling. Assume the following memory-mapped keyboard registers:



Assume the following C function prototypes:

```
// reads a string from the keyboard into buf,
// terminating when CR is read, and returns its length
int gets(char* buf);

// returns a character read from the keyboard
char getc();
```

Further assume:

- That function parameters are passed and values returned in R0.
- KBD_DATA stores the most recently typed character
- KIN is set to indicate data is ready to be read; KIRQ is set to indicate that an interrupt has been requested
- KIE is set to enable interrupts
- CR (13) is the carriage return character

```
kbd: .word  0x00004000
buf: .space 80
.equ CR,    13

main:
   ...
   LDR   A1, ____
   ___   gets
   ...

gets:
   ________
   MOV   V1, ___
   MOV   V2, #0
getsLoop:
   BL    getc
   STRB  A1, [V1], #1
   ADD   ________

   TEQ   A1, #CR
   BNE   getsLoop
   MOV   A1, V2
   ________
   BX    LR

getc:
   ________
getcLoop:
   LDR   V1, ________
   LDRB  A1, [V1, #4]
   TST   A1, #2
   BEQ   getcLoop
   LDRB  ________
   ________
   BX    LR
```