COSC 201 **Sample Exam 1** **Name_____**

1. Define or describe the following:

a. Assembly language
A language for which each instruction is a mnemonic for a corresponding machine instruction, with the exception of a few "pseudo-instructions" which translate into a very few machine instructions. Compilers translate high-level languages into assembly code, which an assembler then translates into machine code.

b. Instruction Set Architecture
The specification of the binary machine instructions that a machine can execute, including the exact layout of the bit pattern for an instruction: the opcodes, any immediate value or address, any registers used. The ISA provides the abstraction on which higher-level languages are built and for which different implementations of the machine can be built.

c. Benchmark Suite
A collection of programs that are used to test the performance of computers. The SPEC benchmark is one example of a benchmark suite developed by a consortium of manufacturers and users.

d. Register indirect (with base) addressing
An address (base) is accessed from a register and is added to an immediate part (offset) to give the address where the desired data item is stored. This is used in ARMv8 in load and store instructions as with
ldur Xa, [Xb, val]
where Xb is the register with the base address that is added to the immediate value val, the data then being stored in register Xa.

e. Register

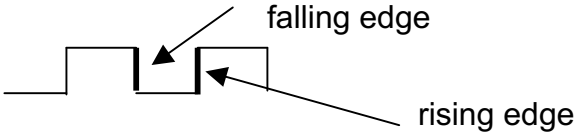
A storage unit for storing a string of bits, typically one "word," in a computer. Usually comprised of flip-flops, one per bit.

f. Hertz

One cycle per second.
This is the unit, usually in mega (millions) or giga (billions) that computer clock rate is measured.

g. Edge-triggered

A storage device such as a register or memory unit is edge-triggered if the state (bit) stored can change only on a clock edge, that is when the clock is changing level from high to low (falling edge-triggered) or low to high (rising edge triggered).



h. DRAM

Dynamic Random Access Memory. A bit is stored as a voltage in a capacitor controlled by a transistor. Since capacitors leak voltage over time, the bits stored need to be refreshed periodically (every few nanoseconds) by reading and rewriting, hence the term dynamic. This makes dynamic RAM slower than the more expensive static RAM (SRAM).

2. Suppose you have the following instruction set mix:

Instr type	CPI	% in program
A	1	50
B	2	25
C	4	25

a. What is the average CPI for this mix?

*CPI = (1 * 0.5) + (2 * 0.25) + (4 * 0.25) = 2.0*

b. If the clock rate is 800 MHz, what is the MIPS for this machine/instruction mix?

*MIPS = 800 MHz / 2.0 CPI = (800 * 10^6 cycles/second) / (2 cycles / instruction) =*
*400 * 10^6 instructions/second = 400 MIPS*

c. Why is MIPS not by itself a good basis for determining the performance of a given machine?
MIPS depends on the instruction mix used in testing -- a program with many type A instructions above will have a higher MIPS -- so it may be compiler dependent. MIPS also depend on the architecture. An architecture with simpler instructions which are fast but do not do as much as instructions on a machine with more complex instructions may have a higher MIPS for a program that runs in the same or slower speed.

3. a. Write ARMv8 assembly code that will execute the following C statement. Assume that the following registers are used to represent the variables.

```
a        in register X19
b        in register X20
c        in register X21

a = (a + b) - (c - 33)
```

```
add   X9,  X19, X20
sub   X10, X21, 33
sub   X19, X9,  X10
```

b. Write ARMv8 assembly code that will execute the following C statements. Assume that the following registers are used to represent the variables:

```
base address of list   in register X19
length of list (len)   in register X20
sum                    in register X21
k                      in register X22

sum = 0;
for(k = 0; k < len; k++)
  sum += list[k];
```

```
      sub   X21, X21, X21          // sum = 0
      sub   X22, X22, X22          // k= 0
for:  sub   X9, X22, X20           // k < len ? (could use subs and b.ge)
      cbz   X9, forend             // if not, end loop
      lsl   X10, X22, 3            // compute X10 = 8*k
      add   X10, X10, X19          // X10 = addr(list[k])
      ldur  X10, [X10, 0]                // X10 = list[k]
      add   X21, X21, X10          // sum += list[k]
      add   X22, X22, 1            // k++
      b     for                    // jump to beginning of loop
forend:
```

4. Consider the following function prototype for C++:

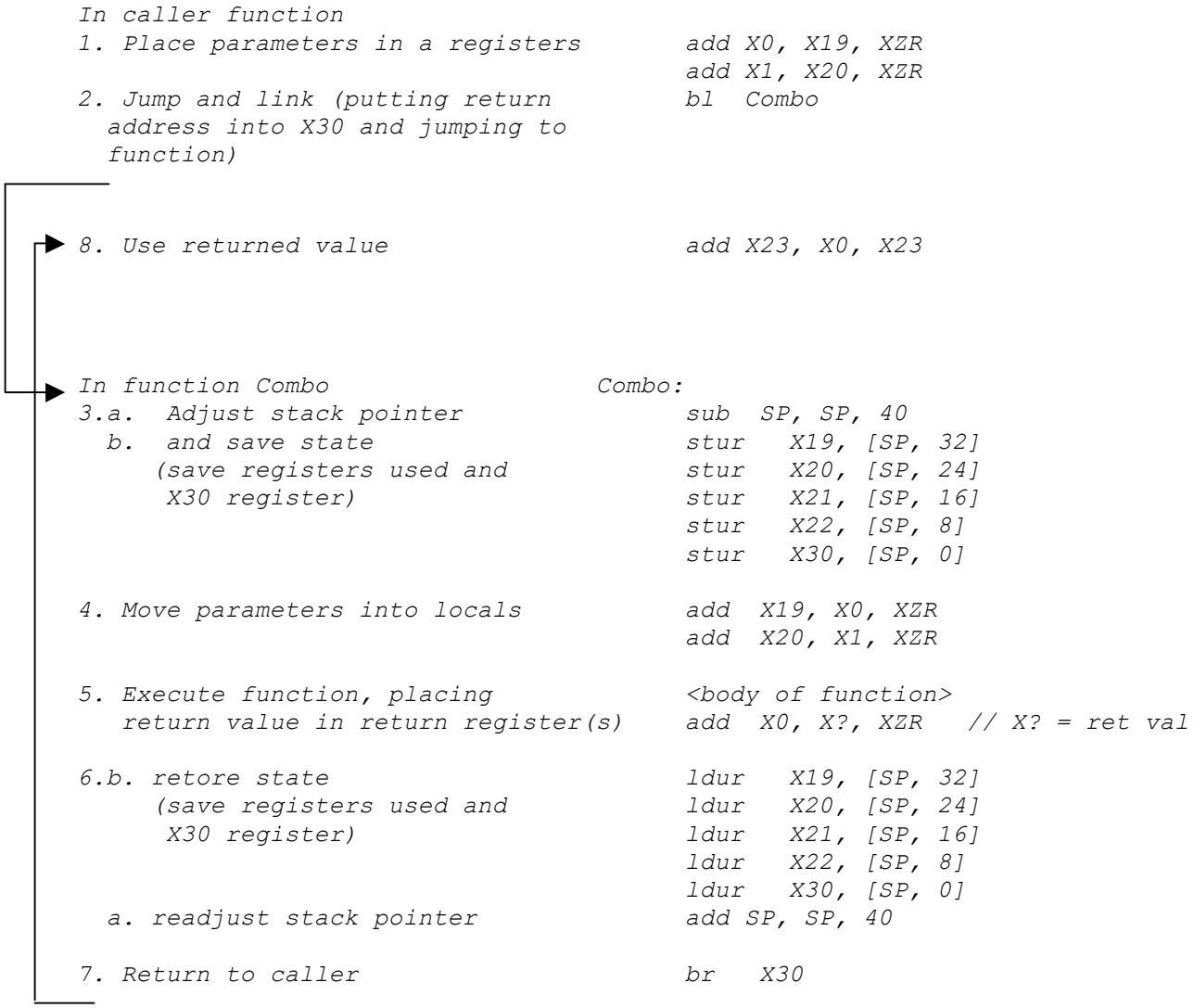
```
int  Combo(int a, int b);
```

Assume that the function `Combo` has six local variables stored in registers X19, X20, X21, X22, X9, and X10, with the usual conventions, where X19 and X20 are used to store the parameters. Assume that the function `Combo` makes calls to other functions. Outline the steps needed to execute the following function call in assembly code, assuming the variables are assigned to registers, `x` in X23, `a` in X19, `b` in X20.

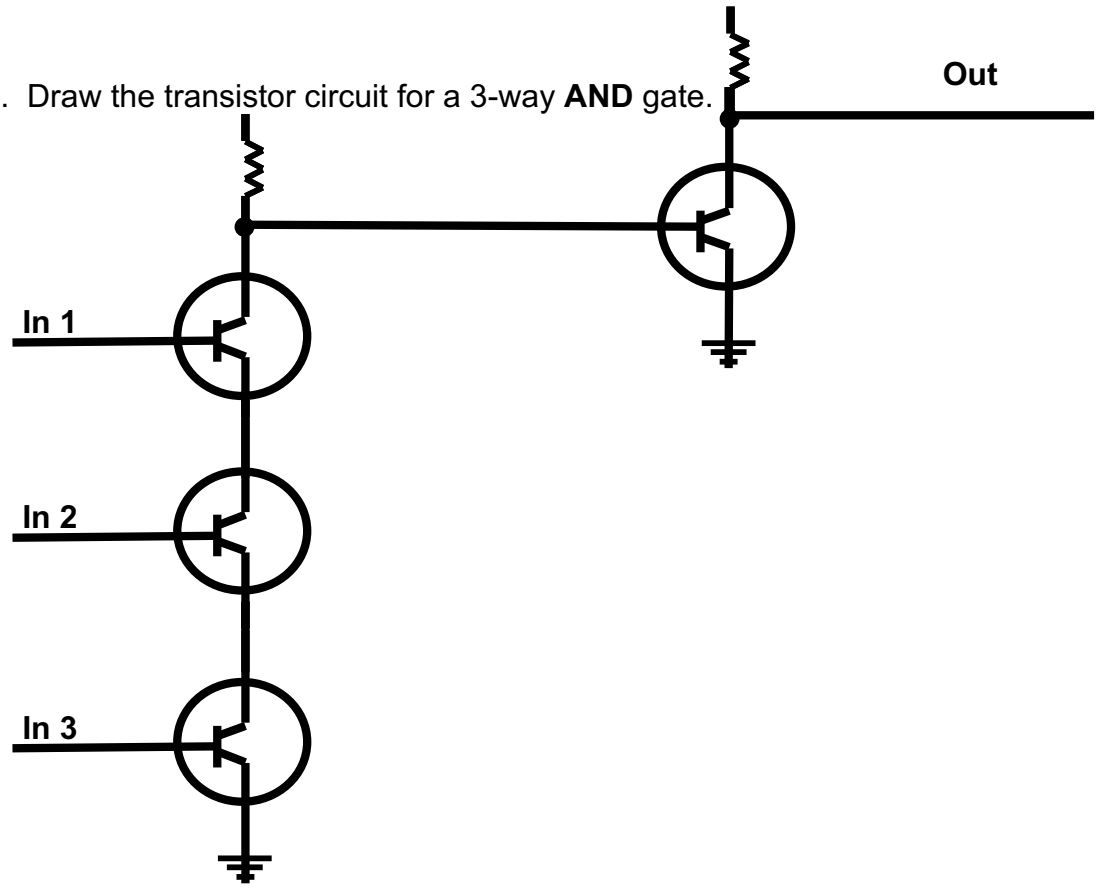
```
x += Combo(a, b);
```

You may answer by explaining the steps in words, or by writing assembly code with comments explaining the purpose of each statement, or a mix of the two. If you are unsure whether something needs to be done, put it in. Include all operations done by the caller function (the function where the above line of code occurs) and the callee function (the function `Combo`) that are needed to make the function call work. Indicate the body of the function `Combo` with a comment <body of function>.

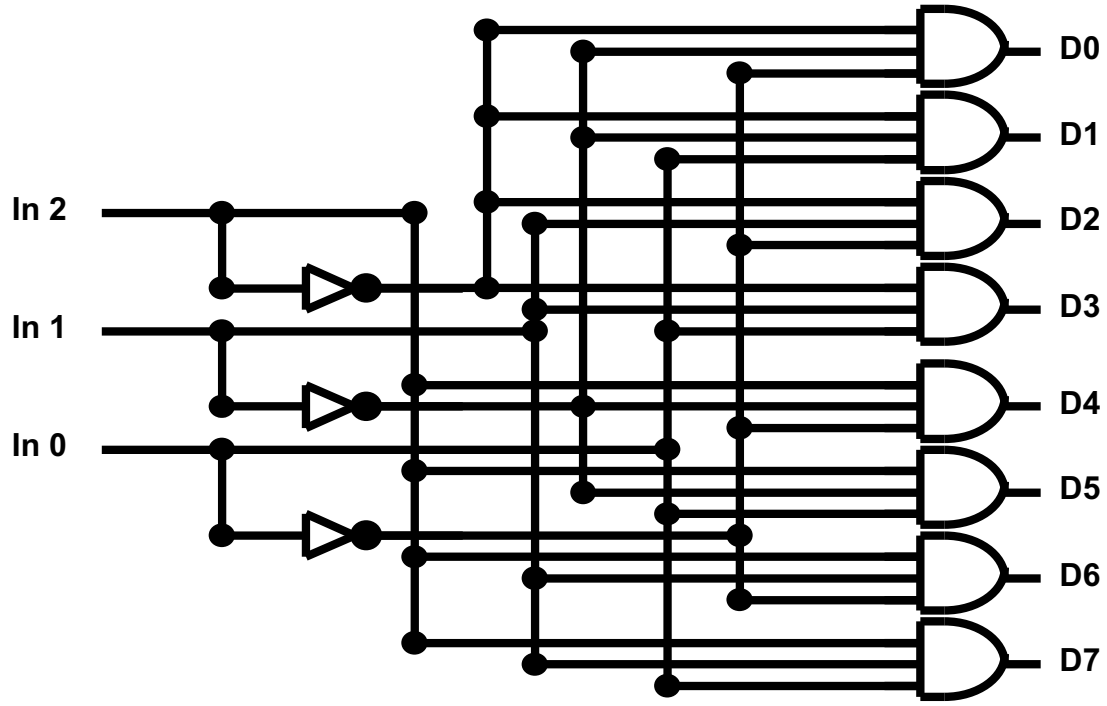
COSC 201 **Sample Exam 1** **Name**_____



5. Draw the transistor circuit for a 3-way **AND** gate.

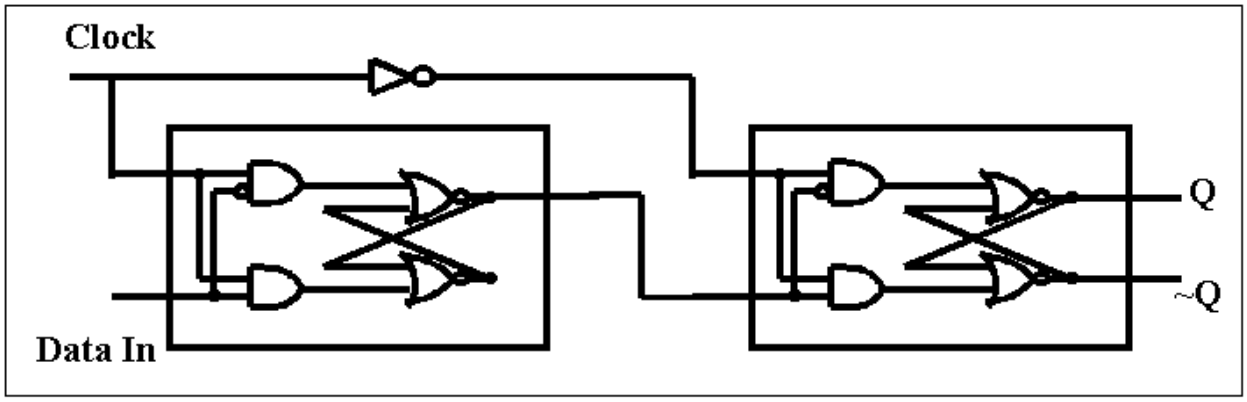


6. Draw the logic gate circuit for a decoder with 3 input lines and the appropriate number of output lines. Where might such a circuit be used?



Used to select which register or memory location to write to.

7. Consider the following diagram.



a. The two parts of the circuit that are boxed are identical. Explain what one of these circuits by itself is and how it works.

Each of these circuits is a D-latch. When the clock is high, whatever value is on the Data In line is stored and is also on the data out line. When the Clock is low, the stored value remains captured on the Out (Q) line, regardless of any changes on the Data In line. The ~Q line has the opposite of the Q line.

b. The complete circuit forms a standard unit. Describe what it is, how it works, and where it might be used in a CPU.

The whole Circuit is a flip-flop, which also stores a single bit value. However, in a flip-flop, the value on the Data-In line is stored during the falling edge of the clock. While the clock is high, the first (master) latch is open and captures the data value. When the clock falls through the falling edge to low, the second (slave) latch (using the negated clock) is open, capturing the signal on the first latch. If the data In signal changes while the clock is low, it will not affect the stored value because the master latch is closed and will not change. If the data value changes while the clock is high, it will change the master latch, but not the slave. It is only the value on the end of the high signal, kept on while the clock falls to low that is captured.

8. Consider the hex number 0x33104000.

a. Write this as a binary number

0011 0011 0001 0000 0100 0000 0000 0000

b. Write this as an integer base ten, assuming two's complement representation. (Write the value as a sum of powers of two, with the appropriate sign, e.g. 2^27 + 2^22 + … + 2^5.)

+ 2^29 + 2^28 + 2^25 + 2^24 + 2^20 +2^14

c. Write this number as a decimal, assuming it is in IEEE-754 floating point representation for 32-bit floats.

Sign is +

Exp is 01100110 in bias 127 this is (2+4+32+64)-127 = -25

Fraction is 0010000010…, so significand is 1.001000001 = 1 + 1/8 + 1/512 = 577/512

Final is + 577 x 2^-34

d. Write the instruction that this represents. (Hint: The opcode 001100 is for the instruction andi)

andi $24, $16, 0x4000 is sufficient. This is also

andi $t8, $s0, 2^15

This was for MIPS. The opcode for ARMv8 would be 11 bits: 00110011000 = 0x198, which is not an opcode for our ARMv8 subset. On an exam you would get a valid opcode with suggestion given in hex, as with the 0x198 here.