1. Briefly describe each of the following and how or where it might be used. A small sketch can be used if it helps the explanation.

a. Instruction Set Architecture (ISA)

The specification of the binary machine language, including how parts of a binary string are interpreted as the following: opcode - indicating type of operation, registers used by the operation, immediate values used by the operation as operands or memory offsets. The assembly language provides human readable instructions that correspond to the machine language, the lowest level of software. Hardware takes instructions as stored in registers to carry out.

b. DRAM (and sketch the circuit for a single bit of storage)

Dynamic Random Access memory – memory values need to be refreshed periodically.

c. Clock cycle. one full repetition in clock signal, as indicated in diagram
d. Flip-flop. One bit memory device that is edge-triggered, made from two D-latches.

e. Level-triggered. A memory device that changes state when the clock is on the high level and remains constant on the low level (or vice versa).

f. Gigahertz. $10^9$ (one billion) cycles per second.
2. Consider the hex number 0xCB180000
a. Write this as a binary number

1100 1011 0001 1000 0000 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. \((-2^{27} + 2^{22} + \ldots + 2^{5})\).)

0011 0100 1110 0111 1111 1111 1111 1111
+1
0011 0100 1110 1000 0000 0000 0000 0000

\(-2^{29} + 2^{26} + 2^{23} + 2^{22} + 2^{21} + 2^{19}\)

c. Write the decimal value that this represents as a 32-bit floating point value using IEEE-754. You may write it as a base-ten value times the appropriate power of two. (for example \(-35 \times 2^{-5}\))

1 10010110 001100\ldots0

sign is negative; exponent is 128+16+4+2-127 = 23

value is 1.0011 = 1 +1/8 + 1/16 = 19/16 = 19 \times 2^{-4}

\(-19 \times 2^{-4} \times 2^{23} = -19 \times 2^{19}\)

d. Write the instruction that this represents. You may represent registers by the X-notation, X0, X1, etc. Remember to group 3 bits, 4 bits, 4 bits to get the opcode hex. Some opcodes: ADD – 0x458, ADDS – 0x558, SUB – 0x658, SUBS – 0x758
(all R-type with bit layout: opcode-register-shamt-register-register)

opcode: 110 0101 1000 = 0x658 so sub
registers are 11000 00000 00000 00000

24. shamt. 0. 0

sub X0, X0, X24
3. Assume a function eval has been written and the assembly code for it is located at the label ”eval;”. The function eval takes a single long integer parameter and returns a long integer. Write the assembly language for the following function. Use comments to indicate how registers are used, like “// X19 is sum” or “// X10 = 8*k”. DO NOT optimize the loop in the function call. Note: this is a function that calls another function!

```c
int countEvalPos(long int[] list, long int length)
{
    count = 0;
    for(long int k = 0; k < length; k++){
        if(eval(list[k]) > 0){
            count++;
        }
    }
    return count;
}
```
3. (continued)

```
sub sp, sp, 40
stur x30, [sp, 0]
stur x19, [sp, 8]
stur x20, [sp, 16]
stur x21, [sp, 24]
stur x22, [sp, 32]

add x19, x0, xzr // x19 is base address of list
add x20, x1, xzr // x20 is length of list
add x21, xzr, xzr // x21 is k = 0 initially
add x22, xzr, xzr // x22 is count = 0 initially
for:
    subs xzr, x21, x20 // k = length
    b.ge endfor // if not k < length, be to endfor
    lsl x9, x21, 3 // x9 = 8*k
    add x9, x9, x19 // x9 is addr of list[k]
    ldur x0, [x9, 0] // x10 = list[k]
    bl eval
    subs xzr, x0, 0 // eval(list[k]) - 0
    b.le endif // if list[k] <= 0 branch to endif
    add x22, x22, 1 // count++
endif:
    add x21, x21, 1 // k++
    b for // next loop

add x0, x22, xzr // result into x0

ldur x30, [sp, 0] // restore values from stack
ldur x19, [sp, 8]
ldur x20, [sp, 16]
ldur x21, [sp, 24]
ldur x22, [sp, 32]
add sp, sp, 40 // reset stack pointer

br x30 // return
```
4. Suppose you have the following instruction set mix:

<table>
<thead>
<tr>
<th>Instr type</th>
<th>CPI</th>
<th>% in program</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>40</td>
</tr>
</tbody>
</table>

a. What is the average CPI for this mix?

\[0.4 \times 1 + 0.2 \times 2 + 0.4 \times 3 = 2.0\text{ CPI}\]

b. If the clock rate is 2 GHz, and the program executes $3 \times 10^9$ instructions, then what is the execution time for this program?

\[
\frac{2\text{ cycles/instruction} \times 3 \times 10^9\text{ instructions}}{2 \times 10^9\text{ cycles/sec}} = 3\text{ sec}
\]

c. How might a manufacturer describe “peak performance” for this machine? Is this misleading?

Use a program with all A type instructions, so 1 CPI or $2 \times 10^9$ instr/sec – 2000 MIPS where the actual program was only at 1000 MIPS
5a. Draw the circuit for a decoder that could be used to set exactly one of four output lines (out0, out1, out2, out3) to one with the others set to zero, based on the value of the input lines. (How many input lines are needed?). Use logic gates and, or, xor, not, as needed.

5b. Where might a decoder be used in circuits that we have discussed?

To select a register to write to in register file
To select memory line to write to.
6. Consider this transistor circuit, where the squares are inputs designated A, B, C and the circle is the output, designated D.

6a. Write the logical expression that this circuit implements (in terms of and, or, not).

\[ D = \neg(A \lor B \lor C) \]

6b. Draw the logic gate symbol that this circuit represents.