1. (10%) Convert the following into single-precision IEEE 754 numbers and display them in hexadecimal.
   
   (a) $-\frac{23}{64}$
   
   (b) $0.0101_2 \times 2^{-130}$

   Answer:
   
   (a) BEB80000
   
   (b) 00028000

2. (6%) Apply DeMorgan’s Theorem to the expression, $A + B + C + AB$, so that the NOT bars do not span more than a single variable (4%). Explain why this function always returns false (2%).

   Answer:
   
   $A + B + C + AB = \overline{A\overline{B}B\overline{C}}$

3. (8%) Consider a 3-input Boolean function that returns true only when exactly two of its inputs are true. (a) Construct its truth table (4%). (b) Implement this function with logic gates AND, OR and NOT (4%).

   Answer:
   
   
   \[
   \begin{array}{ccc|c}
   X & Y & Z & \text{Output} \\
   \hline
   0 & 0 & 0 & 0 \\
   0 & 0 & 1 & 0 \\
   0 & 1 & 0 & 0 \\
   0 & 1 & 1 & 1 \\
   1 & 0 & 0 & 0 \\
   1 & 0 & 1 & 1 \\
   1 & 1 & 0 & 1 \\
   1 & 1 & 1 & 0 \\
   \end{array}
   \]

   (a) $Output = \overline{X}YZ + X\overline{Y}Z + XY\overline{Z}$

4. (6%) TOY ALU uses five control bits to specify its operation (Figure 1(a)). Since there are only seven distinct operations in total, it is possible to specify ALU operations using only three control bits. What change do you have to make to use only three bits to specify TOY ALU operations? What is its downside?

   Answer:
1. Split the adder to adder/subtracter and split shifter to left/right shifter.
2. Add a decoder in front of the original control signal.
3. Reorder the opcode to share the ALU and subtract/shift direction control.

5. (12%) Refer to Figure 1(b) for TOY architecture. Please specify the operations of $MUX_A$, $MUX_B$, $MUX_C$, $MUX_D$, $MUX_E$, $REG_W$, $MEM_W$ and $ALU_{OP}$ during the execution stage for the instructions “load address”, “store indirect” and “branch zero”. For example, the answer for “jump and link” would be $MUX_A = 0, MUX_B = *, MUX_C = *, MUX_D = 1, MUX_E = 01, REG_W = 1, MEM_W = 0, ALU_{OP} = *$.

Answer:

<table>
<thead>
<tr>
<th></th>
<th>$MUX_A$</th>
<th>$MUX_B$</th>
<th>$MUX_C$</th>
<th>$MUX_D$</th>
<th>$MUX_E$</th>
<th>$REG_W$</th>
<th>$MEM_W$</th>
<th>$ALU_{OP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ld</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>ldi</td>
<td>1</td>
<td>0</td>
<td>*</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bz</td>
<td>cond=0</td>
<td>*</td>
<td>0</td>
<td>1</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>*</td>
</tr>
</tbody>
</table>

6. (16%) Write a TOY assembly program which calculates reminder as the following C code. Assume that the data has been defined at the beginning, "POLY DW 0x8320" and "reminder DW 0xFFFF".

```c
for (int bit=0; bit<8; bit++)
    if (reminder&0x01) reminder = (reminder>>1) ^ POLY
    else reminder>>=1
```

Answer:

```assembly
ldi R1 1 ; R1 is always 1
ldi R2 8 ; R2 is counter
ld RA reminder ; RA = reminder
ld RB POLY ; RB = POLY
loop and R3 RA R1 ; R3 = reminder & 1
shl RA RA R1 ; reminder >>= 1
bz R3 else ; if !R3 goto else
xor RA RA RB ; reminder ^= POLY
else sub R2 R2 R1 ; R2--
bp R2 loop ; if R2 > 0 goto loop
st RA reminder ; reminder = RA
```

7. (10%) Answer the following questions:
(a) Is C0A5h a valid hexadecimal constant?
(b) After executing "movsx eax, -4", what is the content of EAX?
(c) The instruction, inc [esi], won’t assemble correctly. Why and suggest a way to fix it.
(d) If AL contains +127 and you add 3 to AL, will the Carry and Overflow flags be set?
(e) The CALL instruction pushed the offset of the CALL instruction on the stack.

Answer:

(a) 0C0A5h
(b) FFFFFFFFC (you will get extra bouns if you point out that "movsx eax, -4" is illegal.)
(c) inc BYTE|WORD|DWORD PTR [esi]
(d) CR=0 and OF=1
(e) The CALL instruction pushed the next offset of the CALL instruction.

8. (12%) Assume that the address of arrayA is 0. (a) what is the content of the memory? Specify it as a byte string and represent each byte in hexadecimal. What are the values of (b) sizeA, (c) TYPE arrayC, (d) LENGTHOF arrayC and (e) SIZEOF arrayC? (f) What is the content of AX after executing "mov al, [arrayC+5]" and "mov ah arrayB[2]?"

arrayA BYTE 24, -2, -12
arrayB SBYTE 0, 16, 26
ALIGN 4
arrayC WORD 1234h, 5678h
WORD 1357h, 2468h
sizeA = ($ - arrayA)

Answer:

<table>
<thead>
<tr>
<th>address</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>18</td>
</tr>
<tr>
<td>0001</td>
<td>FE</td>
</tr>
<tr>
<td>0002</td>
<td>F4</td>
</tr>
<tr>
<td>0003</td>
<td>00</td>
</tr>
<tr>
<td>0004</td>
<td>10</td>
</tr>
<tr>
<td>0005</td>
<td>1A</td>
</tr>
<tr>
<td>0006</td>
<td>*</td>
</tr>
<tr>
<td>0007</td>
<td>*</td>
</tr>
<tr>
<td>0008</td>
<td>34</td>
</tr>
<tr>
<td>0009</td>
<td>12</td>
</tr>
<tr>
<td>000A</td>
<td>78</td>
</tr>
<tr>
<td>000B</td>
<td>56</td>
</tr>
<tr>
<td>000C</td>
<td>57</td>
</tr>
<tr>
<td>000D</td>
<td>13</td>
</tr>
<tr>
<td>000E</td>
<td>68</td>
</tr>
<tr>
<td>000F</td>
<td>24</td>
</tr>
</tbody>
</table>

(a) 0007 * (b) 0Fh (c) 2 (d) 2 (e) 4 (f) 1A13h

9. (5%) What will happen after executing the following code and explain why?

MOV EAX, 10
MOV ECX, 5
L1: MOV ECX, 3
L2: INC EAX
LOOP L2
LOOP L1

Answer:

Infinite loop.
10. (5%) Assume the following initial status of registers,

```
EAX=0000000A  EBX=0000000B  ECX=0000000C  EDX=0000000D
ESI=00000000  EDI=00000000  EBP=0000B000  ESP=00010000
```

what is the content of ESP and EAX after executing the following code?

```
pushad
pop    eax
pop    eax
call   foo ; assume that foo does not have side effects on stack
pop    eax
pop    eax
```

**Answer:**

ESP=0000FFF0 and EAX=00010000

11. (10%) Design a binary multiplier that multiplies two 4-bit numbers as shown in the multiplier block diagram at the beginning, where \( X = X_3X_2X_1X_0 \), \( Y = Y_3Y_2Y_1Y_0 \) and \( Z = X \cdot Y = Z_7Z_6 \ldots Z_0 \), in which \( X_0 \), \( Y_0 \) and \( Z_0 \) are LSBs. You may use the notation \( X[n..m] \) to identify a portion of wires. For example, \( X[3..1] \) means the set of wires, \( X_3X_2X_1 \). (Note that, you do not actually need to use shifters and 8-bit adders to implement the multiplier. If you do, two points may be deducted.)