Examination 1

Name: **CS 2318, Spring 2026** **Score:**

Instructions/directions/notes:

(1) You should attempt to answer all questions.
(2) Closed book and closed notes.
(3) Only writing equipment such as pencils and eraser allowed. No calculators of any kind allowed.
(4) Total points: 90.
(5) Time allowed (in class): 80 minutes.
(6) Last sheet is blank to provide more space for scratch work where needed.

Binary <-> Hex conversion table in case it's helpful

Binary	Hex	Binary	Hex
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

List of observations for use with ***Question 8*** and ***Question 9***

(A) Carry-out is 0 when bits at most-significant-bit position are added.
(B) Carry-out is 1 when bits at most-significant-bit position are added.
(C) Addends have opposite signs.
(D) Addends are both positive, sum is positive.
(E) Addends are both positive, sum is negative.
(F) Addends are both negative, sum is positive.
(G) Addends are both negative, sum is negative.
(H) Carry-in and carry-out at most-significant-bit position are the same.
(I) Carry-in and carry-out at most-significant-bit position are not the same.

1. *(12 points)* Clearly circle **T**(rue) or **F**(alse) for each of the following.

- [T F] Assembly code written to run on a MIPS CPU will as is also run on an INTEL CPU.
- [T F] When writing a program to solve a certain problem using a certain algorithm, we will usually have to write more lines of code if we use assembly language instead of a higher level language.
- [T F] We can only use bits (*i.e.*, 0's and 1's) to represent something *digitally*.
- [T F] The *instruction register* (IR) holds the address of the next instruction to be fetched.
- [T F] The *program counter* (PC) holds a machine language instruction.
- [T F] The present-day predominant finite-state device (transistor) uses a *switching* circuit.

2. *(10 points)* Indicate which item on the right best matches each of those on the left by filling each of the blanks on the right with the corresponding number. (Each blank should have a different number.)

assembly to machine code ①	___ *analog representation*
hardware-software interface ②	___ *assembler*
high-level to assembly code ③	___ *compiler*
represent data discretely with set of finite states ④	___ *digital representation*
represent data continuously with set of infinite states ⑤	___ *instruction set architecture*

3. *(3 points)* Which of the following does <u>not</u> logically belong to ISA (Instruction Set Architecture)? (***Check only 1.***)

☐ How each supported instruction is to be implemented in detail
☐ How many general purpose registers to make available (to programmers)
☐ Which data types to support
☐ Which addressing modes (ways of specifying operands) to support
☐ Which instructions (operations) to support

4. *(5 points)* The range of *unsigned whole numbers* that can be represented using 5 *bits* is *0* through ________, *inclusive*. If we are to represent the same range of *unsigned whole numbers* using *4-state* (instead of *2-state*) devices, the *minimum* number of "*qits*" (quaternary digits) needed is ________.

5. *(3 points)* Check all values below that are out of the range of *32*-bit *2's complement* representation?

☐ $(2^{31} + 1)$
☐ 2^{31}
☐ $(2^{31} - 1)$
☐ $-(2^{31} - 1)$
☐ -2^{31}
☐ $-(2^{31} + 1)$

6. *(12 points)* Fill in the missing entries in the following table, giving four different representations of the <u>*same value in each row*</u>. Use <u>six bits</u> for the binary representations, as many digits (and -) as necessary for decimal. If the value cannot be represented in 6 bits, write "overflow" for the answer.

Decimal	Sign Magnitude	2's Complement	Excess – 34
		010011	
-28			
			001100
	011110		

7. *(3 points)* Express the *unsigned binary* `10.10101` in *hex*.

8. *(5 points)* What is the 8-bit result of adding the following *2's complement* numbers? Indicate if overflow (the result cannot be represented in *8-bit 2's complement*) has occurred and indicate which <u>two</u> of the observations listed on the cover page are applicable and useful for the overflow detection.

```
 [ | | | | | | | |0]  ← show all carry bits on top row
    1 1 0 1 1 0 1 0
 +  0 1 1 1 1 0 0 1
 ------------------
   [ | | | | | | | ]  ← show sum bits on this row
```

Overflow occurred? [**Yes** **No**]

Explain (write the bullet characters for ***2 items*** from cover page list): ________

9. *(3 points)* If the two numbers to be added in ***Question 8*** are *unsigned* numbers, indicate if overflow (the result cannot be represented in *8-bit unsigned*) has occurred and explain how you detect it based on the state(s) of "certain bit(s) in 1 or 2 positions".

Overflow occurred? [**Yes** **No**]

Explain (write the bullet character for ***1 item*** from cover page list): ________________________

10. *(3 points)* Using *Horner's method*, find the *decimal* value corresponding to the *unsigned* ***base-5*** number $\mathbf{2134_5}$.
IMPORTANT: Must show working that *clearly demonstrates use of Horner's method* to get points.

11. *(3 points)* What exactly does `43ABCD12` (a 32-bit pattern shown in hex) represent? (***Check only 1.***)

- ☐ a rather big unsigned whole number
- ☐ a positive 2's complement number
- ☐ a positive IEEE single-precision floating-point number
- ☐ exactly four ASCII characters
- ☐ can't be determined from information given

12. *(4 points)* Order these 2's complement number in binary from 1 to 4 (most –ve to most +ve), <u>WITHOUT</u> resorting to evaluating them in decimal

_____ `11110000001`

_____ `11110000111`

_____ `01110000001`

_____ `11110110001`

13. *(2 points)* Check the 2 parts a computer instruction is made up of when viewed at a high level.

- ☐ opcode
- ☐ operator
- ☐ operands
- ☐ option flag

14. *(4 points)* *Lengthen* the *two's complement* numbers (both more compactly shown in <u>*hex*</u> and are ***already in two's complement*** form) **`34`** and **`EF`** to <u>16 bits</u>.
(<u>NOTE</u>: The values represented in each case should remain the same after lengthening.)

Your 16-bit answer *in* <u>*hex*</u> (for **`34`** in two's complement): ____________________________________.

Your 16-bit answer *in* <u>*hex*</u> (for **`EF`** in two's complement): ____________________________________.

15. *(4 points)* *Shorten* (if possible) the *two's complement* numbers (both more compactly shown in <u>*hex*</u> and are ***already in two's complement*** form) **`34`** and **`EF`** to the *minimum* number of bits needed.
(<u>NOTE 1</u>: The values represented in each case should remain the same after any shortening.)
(<u>NOTE 2</u>: The minimum number of bits needed for the two numbers *need not be the same*.)

Your answer *in* <u>*binary*</u> (for **`34`** in two's complement using minimum number of bits): ___________.

Your answer *in* <u>*binary*</u> (for **`EF`** in two's complement using minimum number of bits): ___________.

16. *(4 points)* Order the following IEEE 754-1985 single precision floating point numbers from 1 to 4, with 1 indicating the smallest (most negative) and 4 indicating the largest (most positive).
<u>TIP</u>: Order them by inspecting/comparing certain part(s) of the representations. You <u>DON'T</u> have to determine the *exact* decimal values represented. Put the **HELPER of Question 17** to good use.

______ `0 11010011 11010011101010010101001`

______ `1 11010011 00101100010101101010110`

______ `0 01010011 11010011101010010101001`

______ `0 01010011 00101100010101101010110`

17. *(10 points)* Represent the *decimal* value **`23.45`** in IEEE 754-1985 ***single*** precision.

HELPER: $(-1)^{S} \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - 127)}$

S	Exponent (8 bits)	Fraction (23 bits)

Your answer (in <u>*hex*</u>): ____.
(To be eligible for potential partial credit, show working below.)

(this last sheet is blank to provide more space for scratch work where needed)