Homework 1 (DAC-102 and DAE-101 Computer Architecture 2026)

Mehta Family School of Data Science and Artificial Intelligence, IIT Roorkee

In all the following questions, assume the RV32 instruction set unless specified otherwise.

1. Subtract each, as a computer would, using binary code (2s complement representation) using registers of size 8. Truncate any bits that do not fit in the register. Subtraction using binary form works in the following manner: (3)

$$\mathbf{X - Y = X + (-Y)}$$

a) 26 – 15
b) 31 – 6
c) 144 – 156

Make sure to check if the result produced is correct. Show your work.

2. Registers a0 and a1 contain the following values:

a0 -> 0xFFFFFFFC
a1 -> 0x00000003

a) What do these numbers represent if stored in twos complement form? (1)

The following instruction is run.

Add a2, a0, a1

What would this addition operation result in if the answer is interpreted as a **b)** signed and **c)** unsigned number? (2)

3. Write the corresponding RISC-V assembly code for the following C statement. Please assume that the variables f, g, h, i, and j are assigned to registers x5, x6, x7, x28, and x29 respectively. Assume that the base address of the arrays A and B are in registers x10 and x11, respectively. (3)

C[21] = A[i−j] + B[g-h];

4. Translate the following C code to RISC-V. Assume that the variables f, g, h, i, and j are assigned to registers x5, x6, x7, x28, and x29, respectively. Assume that the base address of the arrays A and B are in registers x10 and x11, respectively. Assume that the elements of the arrays A and B are **8-byte words**: (3)

B[3] = A[i] + A[j];

5. Assume that the variables f, g, h, i, and j are assigned to registers x5, x6, x7, x28, and x29, respectively. Assume that the base address of the arrays A and B are in registers x10 and x11, respectively. The address stored in A is **0x10000000**.
(3)

```
addi x30, x10, 4
addi x31, x10, 0
sw x31, 0(x30)
lw x30, 0(x30)
add x5,x30, x31
```

What is the value stored in **x5** after these instructions are run?

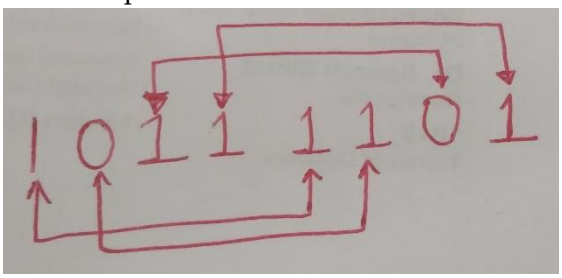
6. Write a minimal sequence of RISC-V assembly instructions for the following C statement that performs the identical operation. Assume x6 = A, and x17 is the base address of C. (2)

A = C[0] << 3;

7. The 32-bit word '**0x34789765**' will be stored at the address location '**0x10000004**' using only the '**sb**'- store byte instruction. You can use the '**li**' instruction to load the numbers into registers and write the program.
(2)
8. Write a program to multiply the numbers -43 and 187. Use the 'Signed' and 'Unsigned' versions of the instructions to multiply the upper significant part of the number. Verify the results and make sure to comment on the difference between the two. (2)
9. Consider a 8-bit number. We examine 4-bits at a time and check whether those bits match 1101. For example, in "1011 1101", this pattern occurs in last four bits. In the number "1011 0100" , this pattern is NOT found in either first four bits or last four bits (although it is found in middle bits, but we do not test that). This test can be extended to any K-bit number, where K is a multiple of 4.

   Assuming RV32, write a program that checks how many times the pattern 1101 occurs in a 32-bit register. The number to check is stored in a0, and the answer should be stored in a1. For example, in the 16-bit number "1101 0011 1101 0001", the pattern 1101 comes two times, so your answer should be 2.
   (4)
10. Write a code to add the number of ones in a 32-bit register. The number is stored in a0, and the answer should be stored in a1.
    (4)
11. Consider a 32-bit number. We can mentally divide the number in two halves (each of 16 bits). Then, we can perform an XOR operation between the bits in the upper half and bits in lower half. The answer will be a 16-bit number (stored in a 32-bit register, with top 16 bits reset to zero). Write a code to perform this operation. The number is stored in a0, and the answer should be stored in a1.

An example of above for an 8-bit number is shown in the figure below.



(4)

12. Load the following words from memory (Minimize the number of instructions):

a. Stored at the address 0x80000200.
b. Stored 0x2000 bytes ahead of the current PC value.
(4 marks)