Assignment 1

Pulak Deb Roy
23241078
Sec - 10

Qus-1:

In RISC-V, we have 8 arguments which are passed through x10 to x17. If a function requires more than 8 arguments, the additional arguements are passed on the stack.

The first 8 arguments are stored in registers. The remaining arguments stored in stack. at stack offsets from the stack pointer. stacks are used due to limited no. of registers, memory managements and many more.

Qus-2:

Here are the key aspects we can understand by examining the opcode field:

① Type of operations
② Instruction format
③ Instruction class
④ Operand Requirements
⑤ control signals.

Qus-3:

The machine will understand the size of the data being loaded into register x9 based on the "funct3" field in the RISC-V instruction.

Qus-4:

The Program counter (PC) is a special register in a CPU that holds the address of the next instruction to be executed.

Importance of a PC:

① Sequential execution: The PC ensures that instructions are executed in order by pointing to the next instruction in memory.

② Control flow management: The PC allows for jumps, branches and function calls by updating its value to point to a different memory address, enabling non-sequential execution of instruction.

③ Program tracking: The PC helps in tracking the current position in the program, which is essential for debugging.

Ans -5:

In RISC-V, branch instruction offsets are specified in terms of 2byte (16-bit) increments. This means that each unit in immediate value represents 2bytes. Since, RISC-V instructions are 4 byte (32-bit) long and aligned to 4byte boundaries, this approach saves bits in the instruction encoding.

In the given machine code, the last 7 bits are "1100011" which indicate an SB-type instruction. Now, extracting the immediate values, we get "0110 1101 0111" which if converted into binary we get 1751. Since the offset is specified in 2 byte unit, the actual offset will be = 1751 × 2

= 3502.

Here, the process describes an efficient encoding that allows the processor to use fewer bits, while still addressing all possible target instruction.

Qus-6:

The "BEQ X0, X0, Label" instruction is called an unconditional jump because "X0" is always zero. Therefore, the condition "X0=X0" is always true, causing the branch to "Label" to always be taken.

Qus-7:

Although we are transferring control, in both cases still we used different instruction for it because —

* Jal = "Jal" known as jump and link used by the caller "Main()" to jump to the calle "addition()" and save the return address in a register x1.

* Jalr = Jalr (Jump and link register) used by the callee "addition()" to return to the caller "Main()" using the address in a register x1, without saving a new return address.

Qus-8(a):

```
If:
    ld  x5, 24(x20)
    ld  x6, 48(x20)
    beq  x5, x6, else

If2:
    ld  x5, 24(x20)
    bne  x5, x0, else2
    addi  x5, x5, 2
    sd  x5, 24(x20)
    beq  x0, x0, exit

else2:
    ld  x6, 48(x20)
    srli  x6, x6, 4
    sd  x6 . 48(x20)
    beq  x0, x0, exit

else1:
    ld  x6, 48(x20)
    slli  x6 . x6, 3
    sd  x6, 48(x20)

exit:
```

Qus-8(b):

For an I-type instruction,

ld x5, 24(x20)

Here,

x20 → rs1 ⇒ 10100

x5 → rd ⇒ 00101

24 → imm ⇒ 000000011000

For S-type instruction:

sd x6, 48(x20)

Here,

rs1 = x20 = 10100

rs2 = x6 = 00110

imm = 48 = 00 00 00 11 0000