CSE 112:
Computer Organization
Duration : 45 Minutes
Date: May 12, 2023

# Quiz II

IIIT-Delhi, Winter 2023
Sujay Deb
Tammam Tillo

## Guidelines

*If you found any ambiguity in any of the questions or there appears to be a lack of information, then write an assumption on the answer sheet to explain your side interpretation of the problem and solve accordingly.*

**Problem I: Instruction Set Architecture and Type of Instruction Classification** **0.5x10 = 5 Points**

***Note**: Students need to answer any 10 choices.*

Classify the below listed instructions in

a) Control Instructions.
b) Memory Instructions.
c) Arithmetic & Logical Instructions.
d) Miscellaneous instructions.

| Instruction (Mnemonic, Operands) | Operation performed | Type of Instruction |
|---|---|---|
| SUB R1, R2, R3 | Subtract the content of R3 from R2 and store the result in R1. | |
| BNE R2, Address | Branch to the location specified by the address, if content of R2 is not equal to zero. | |
| LD R1, Address | Load the content of the specified address in register R1. | |
| ST R4, Address | Store the contents of register R4 to the specified memory address. | |
| HALT | Halt(Stop) the program execution. | |
| ISB | Instruction synchronization barrier(flushes the processor pipeline and all subsequent instructions will be fetched from cache). | |
| LSL R1, #02 | Left shift content of register R1 by 2-bits. | |
| NOP | Performs no operation. | |

Mention the ISA class followed by each instruction(e.g. Accumulator, Register-Memory, Register-Register).

| Instruction (Mnemonic, Operands) | Operation performed | ISA class followed while designing Instruction. |
|---|---|---|
| ADD R7 | Add the content of Accumulator, R7 and store the result in R1 (Accumulator = Accumulator + R7). | |
| SUB R1, R2, R3 | Subtract R3 from R2 and store the result in R1 (R1 = R2 - R3). | |
| ADD R1, R2, [R3] | Add the content pointed by register R3 with register R2 and store the result in R1. | |

**Problem II: Assembly Program Execution** **3x3 + 1 = 10 Points**

Instruction set specified for the below problem.

| Instruction (Mnemonic, Operands) | Shorthand explanation | Operation performed |
|---|---|---|
| ADD R1, R2, R3 | Add Operation. | Add the content of register R2, R3 and store the result into register R1. |
| MOV R1, #Imm | Move(copy) immediate instruction. | Move(copy) the specified immediate value into specified register R1. |
| BL Address | Branch and Link. | Branch to the mentioned address and store the return address(address of next instruction) in the Link register. |
| BGL R1, R2, Address | Branch if greater and link. | Branch to address and also store the return address in Link register, if content of register R1 is greater then content of register R2. |
| BNE R4, R5, Address | Branch if not equal. | Branch to the specified address if the content of registers R4, R5 are not equal. |
| HALT | Halt the program execution. | Stop the program execution immediately. |
| RET | Return from the current subroutine. | Jump to the memory address present in the Link register. |

***Note**: All of the general purpose registers are initialized with their register numbers. For example R0 contains "0", R9 contains "9". And all representations are in decimal format. Please note that there is no stack available to us.*

**Question :** Write the content of registers R6, R7 every time we enter into a subroutine and everytime we exit out from a subroutine.Also mention each content stored in Link register till the program encounters a "HALT" instruction. Also mention how many loop iterations are there.

***Evaluation Note**: There is a pattern of the register content and for the correct pattern of each register(R6,R7,LR) you will be awarded 3-marks.*

- - - - - - - - - - - - - - - - - Assembly program code begins - - - - - - - - - - - - - - - - -

```
BGL R7, R6, label1
BGL R6, R7, label2
HALT
label1:
BGL R5, R4, label3
ADD R6, R2, R3
BGL R7, R6, label4
BL label5
HALT
label4:
MOV R7, #27
MOV R6, #24
RET
label3:
MOV R1, #01
MOV R3, #00
MOV R2, #08
label6:
ADD R3, R1, R3
BNE R2, R3, label6
RET
label5:
MOV R7, #12
RET
label2:
MOV R6, #15
RET
HALT
```

- - - - - - - - - - - - - - - - - Assembly program code ends- - - - - - - - - - - - - - - - -

**Problem III: Assembly Program Encoding and ISA evaluation** **4x2 + 2 = 10 Points**

Write two assembly programs while following two different ISA as mentioned-below and comment on the below specified evaluation parameter.

Algorithm of the program to be converted : Factorial Calculation of a given number.

- - - - - - - - - - - - - - - - - Program pseudo code begins - - - - - - - - - - - - - - - - -

```
temp_number = number_whose_factorial_is_to_be_calculated
multiplied_number = 1
while(temp_number != 0)
        multiplied_number = multiplied_number * temp_number
        // multiply the current number with the old multiplication result starting from factorial_number(to create series of multiplication)
        temp_number = temp_number - 1
        // decrement the factorial_number by 1
Store the result(multiplied_number) to any arbitrary memory location.
```

- - - - - - - - - - - - - - - - - Program pseudo code ends - - - - - - - - - - - - - - - - -

***Note**: Students need to follow the above-mentioned factorial calculation algorithm to write the assembly program.*

| **Instruction Set A** | | **Instruction Set B** | |
|---|---|---|---|
| Instruction (Mnemonic, Operands) | Operation Performed | Instruction (Mnemonic, Operands) | Operation Performed |
| SUB R1, #Imm | Subtract the content of specified register(R1) by immediate. | DCR | Decrement the content of Accumulator by 1. |
| BNZ R1, Address | Branch to address if the content of specified register(R1) is not equal to zero. | JNZ Address | Jump to the specified address if the zero flag is not set. |
| MUL R1, R2 | Multiply reg R1, R2 and store the result into the left_hand_side specified register(R1). | MVI #Imm | Copy the specified immediate value in the Accumulator. |
| MOV R1, R2 | Copy the content of rigth_hand side specified register(R2) into left_hand_side specified register(R1). | MUL R1 | Multiply accumulator with the specified register(R1) and store the result in Accumulator. |
| LD R1, Address | Load the content specified at address into specified register(R1). | MOV [MEM], A | Copy the content of Accumulator to the specified memory location(MEM: memory address). |
| ST R1, Address | Store the content of register R1 to the specified address. | MOV A, [MEM] | Copy the memory content into the accumulator (MEM: memory address). |
| MVI R1, #Imm | Copy the Immediate value into the specified register(R1). | CMP #Imm | Compare the immediate value with the accumulator content and set the zero flag if results are equal otherwise reset the zero flag. |
| | | *MOV reg1, reg2 | Copy the content of specified(reg2) in specified(reg1). |

**Evaluation Parameter:** Which Program needs more number of instructions for the specified program and why.

***Note**: There is no restriction on the number of general purpose registers available in both the ISA. HALT instruction is available in both ISA. And, the number whose factorial is to be calculated is stored at any arbitrary chosen memory location.*

*** Hint** : Specified register can be any general purpose register or Accumulator.