ព្រះរាជាណាចក្រកម្ពុជា
ជាតិ សាសនា ព្រះមហាក្សត្រ

Assignment of Computer Architecture
GROUP: I2-GIC1-B

Name of Students	ID of Students	Score
1.KEO CHANPONLORK	e20220660	…….
2.KEO SIENGHENG	e20221161	…….
3.KUOCH CHYHANG	e20220574	…….
4.KAING SOPHEA	e20220329	…….
5.HONG DYHENGMESA	e20220997	…….

Lecturer: Dr. CHUN Thavorac (Cours)
Mrs. UN Lykong (TP)

Academic Year 2023-2024

I. Question

1. What is the main function of CPU?
2. What does the control unit do?
3. What purpose does a datapath serve?
4. Compare CISC machine to RISC machine.
5. Explain the steps of fetch-decode-execute cycle.
6. Explain the difference between register-to-register, register-to-memory, and memory-to memory instructions.
7. What is a big and little endian?

Answer

1. The main function of a CPU (Central Processing Unit) is to execute instructions that make up a computer program.
2. The control unit (CU) is a critical component of the CPU that manages and coordinates the activities of the computer's hardware.
3. The datapath is a crucial component of the CPU that handles the actual data processing operations.
4. Compare CISC machine to RISC machine:
    - CISC (Complex Instruction Set Computing)
        - Design: Large set of complex instructions.
        - Execution: Instructions can take multiple cycles.
        - Hardware: More complex and power-hungry.
        - Example: Intel x86.
    - RISC (Reduced Instruction Set Computing)
        - Design: Small, optimized set of simple instructions.
        - Execution: Instructions typically execute in a single cycle.
        - Hardware: Simpler, more efficient, and easier to pipeline.
        - Example: ARM architecture.
    - Key Differences
        - Instruction Complexity: CISC is complex; RISC is simple.
        - Execution Speed: CISC is slower per instruction; RISC is faster per instruction.
        - Hardware: CISC is complex; RISC is streamlined.
        - Efficiency: RISC generally has better performance and power efficiency.

5. Explain the steps of fetch-decode-execute recycle:
    a. Fetch:
        - Retrieve Instruction: Get the next instruction from memory using the address in the Program Counter (PC).
        - Store Instruction: Place the instruction in the Instruction Register (IR).
        - Update PC: Increment the PC to point to the next instruction.
    b. Decode:
        - Interpret Instruction: The Control Unit reads the instruction in the IR.
        - Identify Components: Break down the instruction into opcode (operation code) and operands (data or addresses).
        - Generate Signals: Produce control signals to direct other CPU parts.
    c. Execute:
        - Perform Operation: Execute the operation using the Arithmetic Logic Unit (ALU) or other CPU components.
        - Store Result: Write the result to the specified register or memory location.
        - Handle Branches: If it's a jump or branch instruction, update the PC accordingly.
6. Explain the different:
    a. Register-to-Register (Register-Register) Instructions:

        - Description: Operations are performed directly between CPU registers.
        - Example: ADD R1, R2 (adds the contents of registers R1 and R2 and stores the result in R1).
        - Performance: Typically, fast because registers are located within the CPU, reducing the need for accessing slower memory.
    b. Register-to-Memory (Register-Memory) Instructions:

        - Description: Operations involve both a register and a memory location.

- Example: LOAD R1, ADDRESS (loads the value from the specified memory address into register R1).
- Performance: Slower than register-to-register because it involves accessing memory, but it reduces the number of instructions needed to move data.

c. Memory-to-Memory Instructions:
- Description: Operations are performed directly between memory locations.
- Example: MOV ADDRESS1, ADDRESS2 (moves data from one memory location to another).
- Performance: Generally, the slowest because both operands involve memory access, which is much slower than accessing registers.

d. Summary
- Register-to-Register: Fast, uses CPU registers for operations.
- Register-to-Memory: Intermediate speed, involves one register and one memory location.
- Memory-to-Memory: Slowest, involves operations directly between memory locations.

7. The explanation:
- Big Endian is an approach to storing data in which the most significant byte (MSB) is stored at the smallest memory address. Example: For a 32-bit hexadecimal number 0x12345678, in Big Endian format, the byte order would be 12 34 56 78, with 12 stored at the lowest memory address.
- Little Endian is an approach to storing data in which the least significant byte (LSB) is stored at the smallest memory address. Example: For a 32-bit hexadecimal number 0x12345678, in Little Endian format, the byte order would be 78 56 34 12, with 78 stored at the lowest memory address.

II. Exercise

1. How many bits would you need to address a 2M×32 memory if
   a. The memory is byte-addressable

- $log_2(2M) = 21$
- Each word has 32 bits which equal to 4 bytes ➔ $log_2(4) = 2$
- 21 + 1 = 22 bits

⇨ we need 22 bits to address 2M x 32 memory if memory is byte-addressable

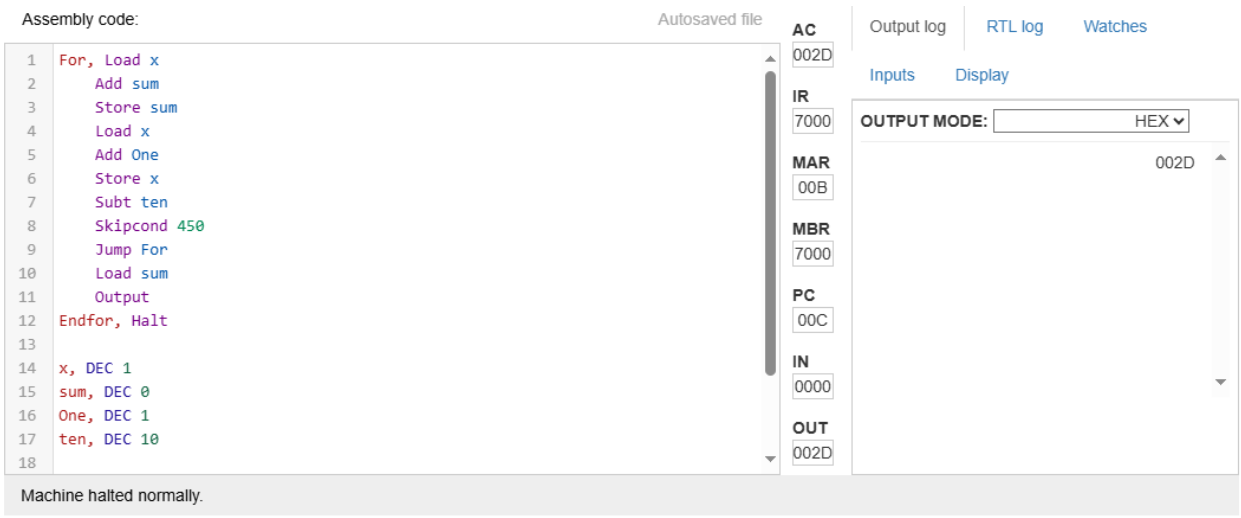
b. The memory is word-addressable

2 M words which has 32 bits, we need to $log_2(2M) = 21$

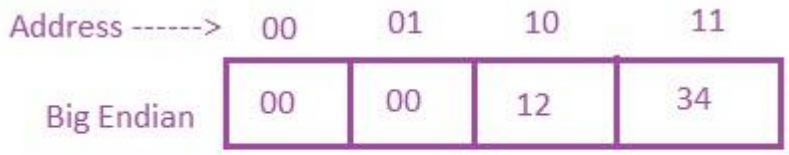
⇨ we need 21 bits to address 2M x 32 memory if memory is word-addressable

2. Write the following code segment in MARIE assembly language:
   Sum = 0;
   for X = 1 to 10 do
   Sum = Sum + X;

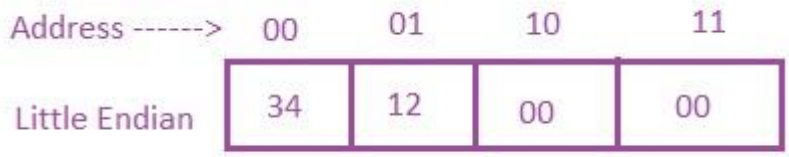
   ⇨ The Code:



3. Assume you have a machine that uses 32-bit integers and you are storing the hex value 1234 at address 0:
   a. The hex value 1234 at address 0 will store on big endian machine are shown in below diagram



b. The hex value 1234 at address 0 will store on little endian machine are shown in below diagram



c. Little endian is more efficient because the additional information simply needs to be appended. With big endian, the "12" and "34" would need to shift to maintain the correct byte ordering.

4. The memory unit of a computer has 256K words of 32 bits each. The computer has an instruction format with 4 fields: an opcode field; a mode field to specify 1 of 7 addressing modes; a register address field to specify 1 of 60 registers; and a memory address field. Assume an instruction is 32 bits long. Answer the following:
   a. The mode field size: 3 bits
   b. The register field size: 6 bits
   c. The address field size: 18 bits
   d. The opcode field size: 32-(3+6+18) = 5 bits