IGNOU | BCA

2023-2024

FREE

SOLVED

ASSIGNMENT

All In One

MCS-012

Computer Organization and Assembly Language Programming

Prepared by

ISM

730-339-3555

www.ignoustudymentor.com

@ism_ignoustudymentor_

ISM - IGNOU STUDY MENTOR

Youtube Channel and

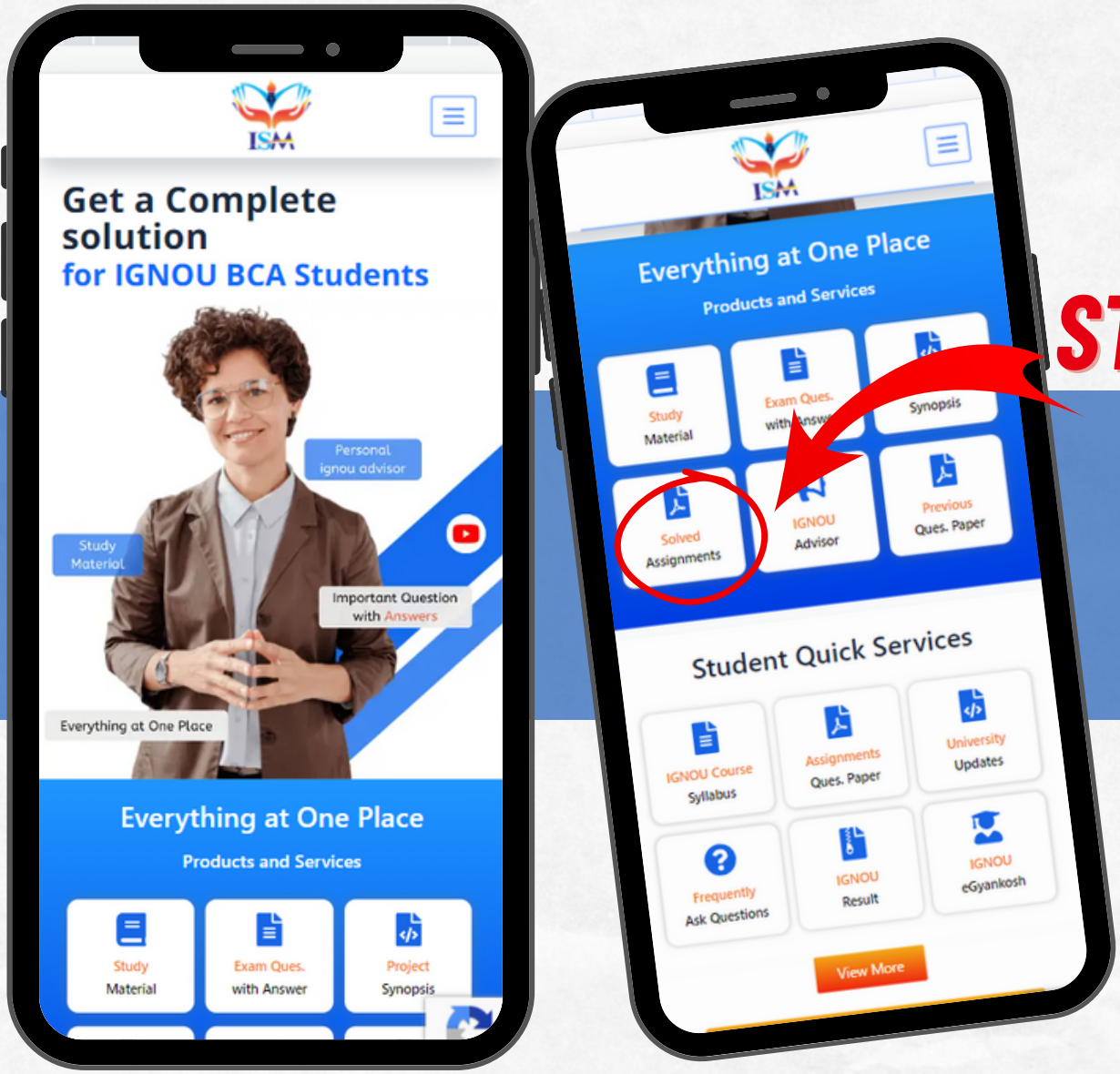
Get a Complete Solution

for IGNOU Students

www.ignoustudymentor.com

STEP 1

www.ignoustudymentor.com



STEP 2

MCS-012

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

[SEM-2]

Ques.Q3.(a) A single-core uniprocessor system has 8 General purpose registers. The machine has RAM ofsize 64K memorywords. The size of every general-purpose register and memory word is 16 bits. The computer uses fixed-length instructions of size 32 bits each. An instruction of the machine can have two operands. One of these operands is a direct memory operand and the other is a register operand. An instruction of a machine consists of bits for operation code, bits for memory operand and bits of register operand. The machine has about 128 different operation codes. The special purpose registers, which are other than general purpose registers, are- Program Counter (PC), MemoryAddress Register (MAR), Data Register (DR) and Flag registers (FR). The first register among the general-purpose registers can be used as Accumulator Register. The size of Integer operands on the machine may be assumed to be equal to the size of the accumulator register. To execute instructions, the machine has another special register called Instruction Register (IR) of size 32 bits, as each instruction is of this size. Perform the following tasks for the machine. (Make and state suitable assumptions, if any.)

MCS-012

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

[SEM-2]

Ans. (i) Design suitable instruction formats for the machine. Specify the size of different fields that are needed in the instruction format. Also, indicate how manybits of the instructions are unused for this machine. Explain your design of the instruction format. Also, indicate the size of each register.

(i) Instruction Format: Based on the given information, we can design the instruction format as follows:- Operation Code (OpCode): 7 bits (to accommodate 128 different operation codes)
- Memory Operand: 9 bits (to address up to 512 memory locations)
- Register Operand: 4 bits (to address up to 16 general-purpose registers)

As the machine uses fixed-length instructions of size 32 bits, the remaining 12 bits can be left unused.

Register Size: Each general-purpose register and memory word is 16 bits, so the size of each register will be 16 bits. The size of the accumulator register will also be 16 bits since it is the first general-purpose register.

| Opcode (7 bits) | Register Operand (4 bits) | Memory Operand (9 bits) |

(ii) Demonstrate two valid instructions of the machine; put some valid data values in registers and memory locations and show these two instructions.

MCS-012

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

[SEM-2]

TwoValid Instructions: Let's assume the following data values:
- Register 1 (Accumulator Register) contains the value 10
- Memory location 100 contains the value 5

Example Instructions: 1. ADDR1, M100
- OpCode: 0010001 (example value)
- Memory Operand: 100
- Register Operand: 0001 (R1

This instruction adds the contents of Memory location 100 (value 5) and Register 1 (value 10) and stores the result in the Accumulator Register.

2. MOVM200,R1
- OpCode: 0100010 (example value)
- Memory Operand: 200
- Register Operand: 0001 (R1)

This instruction moves the value from Register 1 (value 10) to Memory location 200.

MCS-012

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

[SEM-2]

Ques. (iii) Assuming that the instructions are first fetched to the Instruction Register (IR), the memory operand is brought to the DR register and the result of an operation is stored in the Accumulator register; write and explain the sequence of micro-operations that are required to fetch and execute an addition instruction that adds the contents of the memory and register operands of the instruction. The result is stored in the accumulator register. Make and state suitable assumptions, if any.

Ans. Sequence of Micro-operations for Addition Instruction:

1. Fetching:

- The Program Counter (PC) is incremented to point to the next instruction.
- The instruction at the memory location pointed by PC is fetched and stored in the Instruction Register (IR).

2. Decoding:

- The OpCode field in the IR is decoded to determine the specific operation to be performed (in this case, addition).
- The Memory Operand field in the IR is used to determine the memory location to be accessed (in this case, M100).
- The Register Operand field in the IR is used to determine the register to be used (in this case, R1).

3. Memory Access:

-The Memory Address Register (MAR) is loaded with the memory location specified in the Memory Operand field (M100).

- The memory content at the address stored in MAR (value 5) is fetched and stored in the Data Register (DR).

4. Execution:

- The value in DR (5) is added to the value in the Accumulator Register (10).
- The result (15) is stored back in the Accumulator Register.

5. Store Result:

- The value in the Accumulator Register (15) is stored back in the memory location specified by the Memory Operand field (M100).

(b) Assume that you have a machine, as shown in section 3.2.2 of Block 3 having the micro-operations given in Figure 10 on page 62 of Block 3. Consider that R1 and R2 both are 8-bit registers and contain 11100111 and 00111100 respectively. What will be the values of select inputs, carry-in input, and the result of the operation (including carry-out bit) if the following micro-operations are performed? (For each micro-operation you may assume the initial value of R1 and R2 as given above).

(i) Decrement R1

(ii) Add R1 and R2 with Carry

(iii) Exclusive OR of the registers R1 and R2

(iv) Shift right R1

MCS-012

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

[SEM-2]

(i) Decrement R1: The decrement operation decreases the value of R1 by 1. Since R1 initially contains 11100111, decrementing it will result in 11100110. The select inputs and carry-in input are not relevant for this operation.

(ii) Add R1 and R2 with Carry: Adding R1 and R2 with carry means adding the values of R1 and R2 along with an additional carry input. The initial values of R1 and R2 are 11100111 and 00111100 respectively. The carry-in input can be assumed to be 0. Performing the addition:

```
  11100111
 +00111100
 ----------
  100101011
```

The result is 10010101, and the carry-out bit is 1. The select inputs are not relevant for this operation.

(iii) Exclusive OR of the registers R1 and R2: Performing an exclusive OR (XOR) operation on R1 and R2 will result in a new value. The initial values of R1 and R2 are 11100111 and 00111100 respectively. Performing the XOR:

```
    11100111
XOR00111100
----------
    11011011
```

The result is 01110011.

(iv) Shift right R1: Shifting the bits of R1 one position to the right will result in a new value. The initial value of R1 is 11100111. Performing the shift right:

```
11100111
>> 1
--------
01110011
```

The result is 01110011

Therefore, the values of the select inputs, carry-in input, and the result of the operation for the given micro-operations are as follows:

(i) Decrement R1: Select inputs and carry-in input not relevant, result = 11100110.

(ii) Add R1 and R2 with Carry: Select inputs not relevant, carry-in input = 0, result = 10010101 with carry-out bit = 1.

(iii) Exclusive OR of the registers R1 and R2: Select inputs and carry-in input not relevant, result = 11011011.

(iv) Shift right R1: Select inputs and carry-in input not relevant, result = 01110011.

(c) Consider that an instruction pipeline has three stages namely instruction fetch and decode (FD), Operand Fetch (OF) and Instruction Execute and store results (ES). Draw an instruction pipeline diagram showing the execution of five sequential instructions using this pipeline. What are the problems with this instruction pipelining?

The instruction pipeline diagram for the five sequential instructions would look like this:

```
Instruction 1: FD-> OF-> ES
Instruction 2:         FD->OF->ES
Instruction 3:               FD->OF->ES
Instruction 4:                     FD->OF->ES
Instruction 5:                           FD->OF->ES
```

MCS-012

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

[SEM-2]

- The instruction stages (FD, OF, ES) are shown for each instruction as they progress through the pipeline.
- The pipeline begins with instruction 1 in the Fetch/Decode (FD) stage at the first clock cycle.
- In each subsequent clock cycle, the instructions move to the next stage.
- Once an instruction reaches the Execute/Store Results (ES) stage, it completes and exits the pipeline.

Problems with this Instruction Pipelining: Pipeline Hazards

Structural Hazards: These occur when multiple instructions need to use the same hardware component simultaneously. For example, if the Operand Fetch (OF) stage requires memory access while another instruction is already accessing memory.

Data Hazards: These occur when instructions depend on the results of previous instructions. In a pipeline, if an instruction in the OF stage depends on the result of an instruction in the ES stage, there will be a data hazard that requires resolving.

Control Hazards: These occur due to branching instructions that change the flow of execution. The pipeline needs to deal with branch prediction and instruction flushing when a branch is mispredicted.

Pipeline Stall or Bubbles: Pipeline stalls occur when a stage of the pipeline cannot proceed due to a hazard. This leads to empty pipeline stages or "bubbles," which reduce the efficiency of the pipeline.

MCS-012

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

[SEM-2]

Data Forwarding and Delayed Write:

Data forwarding is required to pass the results of an instruction directly to a subsequent instruction that depends on it. Without proper forwarding, stalls may occur. Delayed write refers to waiting to update a register or memory until the result is certain. This can lead to inefficiencies in the pipeline. Instruction Mix and Dependency:

(d) Explain the functioning of the Wilkes Control Unit. Also, explain the format of the control memory with the help of a diagram.

The Wilkes Control Unit, also known as the Wilkes Control Store, is a part of the control unit in a computer's architecture. It's named after its developer, Maurice Wilkes, who designed the control unit for the Electronic Delay Storage Automatic Calculator (EDSAC), one of the earliest stored-program computers.

Functioning of the Wilkes Control Unit:

The Wilkes Control Unit is responsible for orchestrating the execution of instructions in a computer's central processing unit (CPU). It interprets the instruction fetched from memory and generates control signals that coordinate the various components of the CPU to perform the necessary operations for the instruction's execution. These control signals activate specific paths within the CPU, such as the arithmetic logic unit (ALU), registers, memory, and input/output devices, to perform the required tasks.The key feature of the Wilkes Control Unit is that it uses a "control memory" to store micro-instructions or micro-operations. Each micro-instruction corresponds to a specific action that the CPU needs to take to execute an instruction. These actions might include reading from or writing to registers, performing arithmetic or logic operations, accessing memory, and controlling the flow of execution.

MCS-012

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

[SEM-2]

Format of the Control Memory: The control memory in the Wilkes Control Unit typically consists of a series of storage locations, each holding a micro-instruction. Each micro-instruction is made up of control bits that activate or deactivate specific components and operations within the CPU. The format of a typical micro-instruction in the Wilkes Control Store might include the following fields:

1. Opcode: An opcode field specifies the type of operation or action to be performed. It determines what task the micro-instruction will carry out.

2. Control Signals: These fields contain control bits that activate or deactivate various components in the CPU, such as registers, ALU, memory, etc. Each control signal corresponds to a specific function or operation.

3. Next Address: This field indicates the memory address of the next micro-instruction to be fetched. It controls the sequence of execution and supports branching and looping.

Representing the format of a micro-instruction in the Wilkes Control Store:

```
+------+-------------+------+
Opcode | Control Signals (Bits) | Next Address |
+------+-------------+------+
```

The Wilkes Control Unit operates by sequentially fetching micro-instructions from the control memory based on the opcode of the current instruction being executed. The control signals in the micro-instruction then trigger the appropriate actions and operations within the CPU to execute the instruction.

MCS-012

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

[SEM-2]

(e) Explain the characteristics of RISC? Also, explain the RISC pipelining.

RISC, which stands for Reduced Instruction Set Computer, is a type of computer architecture that emphasizes simplicity and efficiency in instruction execution. RISC architectures are designed to have a smaller set of simple and basic instructions that can be executed in a single machine cycle. These architectures contrast with CISC (Complex Instruction Set Computer) architectures, which have a larger and more complex set of instructions that can take multiple machine cycles to execute.

Characteristics of RISC:

1. Simplified Instruction Set: RISC architectures have a reduced and straightforward set of instructions, typically around 100 or fewer instructions. Each instruction performs a simple task, making them more easily decoded and executed in a single clock cycle.

2. Fixed-Length Instructions: RISC instructions are usually of fixed length, which simplifies the decoding process and allows for more predictable execution times.

3. Load-Store Architecture: RISC architectures commonly employ a load-store memory model, where data can only be loaded from memory to registers or stored from registers to memory. Arithmetic and logic operations are performed on registers, not directly on memory.

4. Register-Register Operations: RISC architectures heavily emphasize register-register operations, meaning that most operations are performed between registers. This reduces memory access time and improves performance.

MCS-012

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

[SEM-2]

5. Single-Cycle Execution: RISC instructions are designed to execute in a single clock cycle, making them faster compared to CISC instructions that may require multiple cycles for complex operations.

RISC Pipelining:

Pipelining is a technique used in computer architectures to improve instruction throughput and overall performance. RISC architectures are particularly amenable to pipelining due to their simple instruction set and regular format. The RISC pipeline is typically divided into multiple stages, each handling a different aspect of instruction execution. These stages can include:

1. **Instruction Fetch (IF):** Fetches the instruction from memory.
2. **Instruction Decode (ID):** Decodes the fetched instruction to determine the operation to be performed and the operands involved.
3. **Execute (EX):** Executes the operation, which may involve arithmetic, logic, or other instructions.
4. **MemoryAccess (MEM):** Accesses memory for load and store instructions.
5. **Write-Back (WB):** Writes the result of the executed instruction back to a register.

Follow :

ism_ignoustudymentor_

Join Telegram Group :

ism_ignoustudymentor

Youtube Channel

ISM - IGNOU Study MENTOR

Website

www.ignoustudymentor.com

730-339-3555