Computer Architecture

CSE-3503 | MID Note | sugestion of Sheikh Sir Habib Sir

prepared by:-
Sonowar Mahabub
C201032
Cell: 01522564157

Q1] Why is it important for us to Study Computer Architecture?

Ans: Computer Architecture is a set of rules and methods that describe the functionality, organization and implementation of computer systems. In a nutshell, CA is basically the set of rules that control how hardware and software interact together.

Computer Architecture can Help you more than you think. Reason why should learn CA -

- you will likely use it for the rest of your life.
- CA is one of the most fundamental subjects in Computer Science. As without Computer, Computer Science would not exist.
- you need to understand how the instructions and operations actually work and interact together to make your software better; because whatever you do, no matter what, it is on top of CA.
- Computer Architecture will help you design, develop and implement applications that are better, faster, cheaper, more efficient and easier to use because you will be able to make informed decissions instead of guessing estamating and assuming.
- There will be a time when to debug a problem you will have to understand the hardware.

by-
Sonowar Mahabub
C201032

Q2| Which One Is More Better bet^n RISC and CISC?

Ans: RISC, which is an acronym for Reduced Instruction Set Computer and CISC, short for Complex Instruction Set computer, refer to the category of the processor, or more accurately, the instruction set architecture (ISA).

The main idea behind CISC processors is that a single instruction can be used to do all of the loading, evaluating and sorting operations. The goal of the CISC approch is to minimise the number of instructions per programme.

The primary goal of RISC processors is make hardware simpler by using an instruction set that's composed of a few basic steps for loading, evaluating and sorting operations.

Different bet^n RISC and CISC one given in the following table-

RISC		CISC
Emphasis on Software	a	Emphasis on Hardware
Small number of fixed length instructions	b	Large number of instructions
Simple, standardized instructions	c	Complex, variable-length instructions
Single clock cycle instructions	d	Instructions can take several clock cycles.
Heavy use of RAM	e	More efficient use of RAM
Low cycles per second with large code sizes	f	Small code sizes with high cycles per second

Although the above showcases differences bet^n the

two architectures, the main difference betⁿ RISC and CISC is the CPU time taken to execute a given program. RISC architecture will shorten the execution time by reducing the average clock cycle per one instruction.
Thus, it can be said RISC is better as it also can be designed quickly than CISC processors due to its simple architecture.

by- Sonowar Mahabub

Q3 Which is better/important betⁿ LSI and VLSI?

Ans: LSI stands for Large-Scale Integration and VLSI stands for Very Large-Scale Integration.

Integration	Technology	Typical Number of devices	Typical Functions
LSI	Bipolan and MOS	100–10,000	ROM and RAM
VLSI	CMOS (most commonly)	10,000–5,000,000	processors

Table → 1.2

From the above table, we can see the typical number of devices is more in VLSI than LSI and also VLSI is the current level of computer microchips miniaturization. So, VLSI is obiously better than LSI.

Q4 Defination

Computer Architecture: See in Q1 (page-1)

Computer Organization: It is how operational attributes are linked together and contribute to realizing the architectural specification. It deals with a structural relationship. ALU, CPU, memory are the significant components of Computer Organization.

Relative Performence: It is the ratio bet^n two computer (suppose, A and B)'s performence.

$$R.P = \frac{\text{performence}_A}{\text{performence}_B} = n$$

∴ Machine A is n times faster than B.

Remember,

$$CPI = \frac{\text{CPU clock cycles for the program}}{\text{Instruction Count}} = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{\text{Instruction Count}}$$

$$MIPS = \frac{\text{Instruction Count}}{\text{Execution Time} \times 10^6} = \frac{\text{Clock Rate}}{CPI \times 10^6}$$

Example Consider computing the overall CPI for a machine A for which the following performance measures were recorded when executing a set of benchmark programs. Assume that the clock rate of the CPU is 200 MHz.

Ex-01

Instruction category	Percentage of occurrence	No. of cycles per instruction
ALU	38	1
Load & store	15	3
Branch	42	4
Others	5	5

Assuming the execution of 100 instructions, the overall CPI can be computed as

Example Suppose that the same set of benchmark programs considered above were executed on another machine, call it machine B, for which the following measures were recorded.

Ex-02

Instruction category	Percentage of occurrence	No. of cycles per instruction
ALU	35	1
Load & store	30	2
Branch	15	3
Others	20	5

What is the *MIPS* rating for the machine considered in the previous example (machine A) and machine B assuming a clock rate of 200 MHz?

Q5] Perfomence Measures [Math of page 7, 8]

Ex-01 We know that,

$$CPI = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{\text{Instruction Count}}$$

Here, Instruction count = 100

$$= \frac{38\times1 + 15\times3 + 42\times4 + 5\times5}{100} = 2.76$$

Ex-02 For machine A,

$CPI_a = 2.76$ [—— from Ex-01]

Here, Clock Rate = 200 MHz = 200×10^6 Hz

$$MIPS_a = \frac{\text{clock rate}}{CPI_a \times 10^6} = \frac{200\times10^6}{2.76\times10^6} = 70.24$$

For machine B,

$$CPI_b = \frac{35\times1 + 30\times2 + 15\times3 + 20\times5}{100} = 2.4$$

$$MIPS_b = \frac{\text{Clock Rate}}{CPI_b \times 10^6} = \frac{200\times10^6}{2.4\times10^6} = 83.67$$

Thus $MIPS_b > MIPS_a$

Note that, MIPS → Million instructions-per-second

CPI → Clock Cycle-per-second

by-
Sonowar Mahabub
C201032
cell: 01521564157

Q6) Explanation of different types of parameters

CPI: Cycle Per Instruction or, Clock Per Instruction (CPI) is the number of computer clock speed cycles (alternating current pulse) that occurs while a computer instruction is being executed (performed by the computer processor). The number of cycles per instruction can be reduced by using pipelining. In some superscalar processor, more than one instruction can be performed during a single clock cycle. Clearly, How many cycles an instruction takes in order to be executed.

$$CPI = \frac{\text{CPU clock cycles for the program}}{\text{instruction count}}$$

It is known that the instruction set of a given machine consists of a number of instruction categories: ALU, load, store, branch and so on. So, the overall CPU can be computed as-

$$CPI = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{\text{Instruction Count}}$$

by Sonowar Mahabub C201032

MIPS: it is short for million of instruction-per-second (the rate of instruction execution per unit time), MIPS is the approximate number of instructions a CPU can execute in one second. It helps in the calculation of CPU processor speed (cycles per second), CPI (average clock cycles per instruction) and Execution time. It handle when the amount of work is large. MIPS is defined as-

$$MIPS = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Clock Rate}}{CPI \times 10^6}$$

For example, The intel 80386 (386) computer processor was capable of performing more than five million instructions every second. 5 MIPS.

Q7] Read Operation of Memory:

Ans: To perform a memory read operation, the MDR and MAR are used as follows -

① The address of the location from which the word is to be read is loaded into MAR.

② A read signal is issued by CPU.

③ The required word will be loaded by the memory into the MDR ready for use by the CPU.

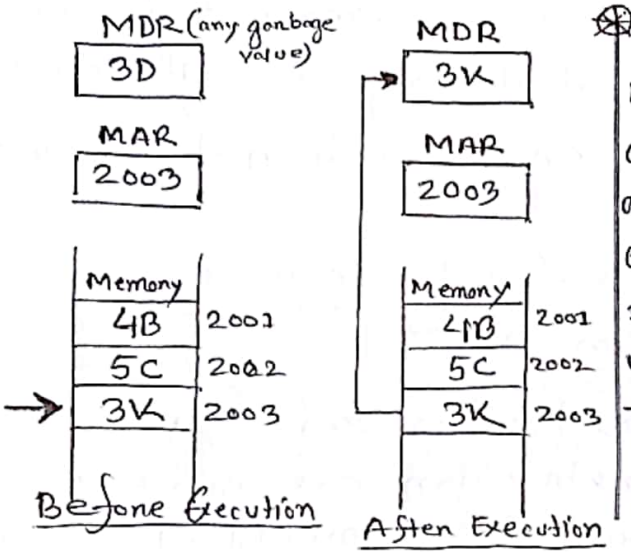


Fig: Memory Read Operation

In the diagram, MDR can contain any garbage value and MAR is containing 2003 memory address. After the execution of read operation, the data of memory location 2003 will be read and the MDR will get updated by the value of the 2003 memory location.

by -
Sonowar Mahabub
C202032

Extra

Note that

MAR: Memory Address Register is the address register which is used to store the address of the memory location, where the operation is being performed.

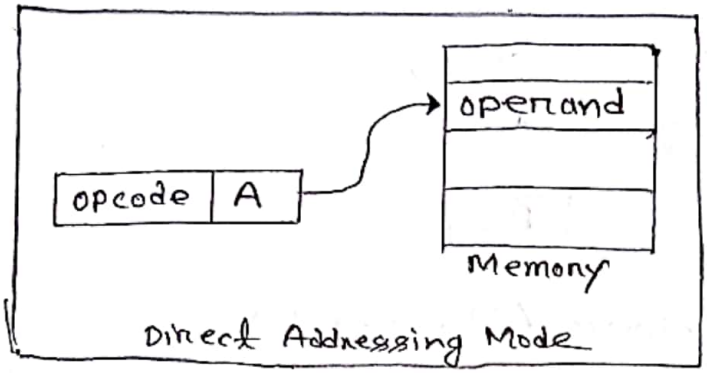
MDR: Memory Data Register is the data register which is used to store the data on which the operation is being performed.

Q8] Addressing Modes:

① **Direct Addressing Mode:** In this addressing mode,

- The address field of the instruction contains the effective address of the operand.

- only one reference to memory is required to fetch the operand.
- It is also called as absolute addressing mode.



Example: ADD X will increment the value stored in the accumulator by the value stored at memory location X.

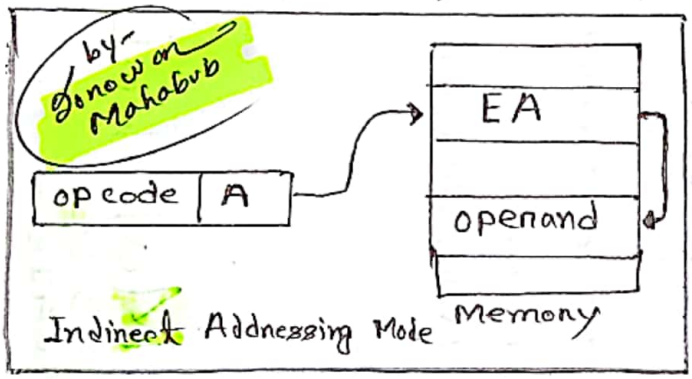
① AC ← AC + [X]

② load 1000, Ri Ri ← M[1000]

② Indirect Addressing Mode:

In this addressing mode,

- The address field of the instructions specifies the address of memory location that contains the effective address of the operand.
- Two references to memory are required to fetch the operand.



Example: ADD X will increment the value stored in the accumulator by the value stored at memory location specified by X.

AC ← AC + [[X]]

→ Based on the ability of Effective address, IDM is two kinds-

ⓐ Register Indirect: one register and one memory reference is required to access data.

ⓑ Memory Indirect: Two memory reference is required to access data.

③ Relative Addressing Mode:

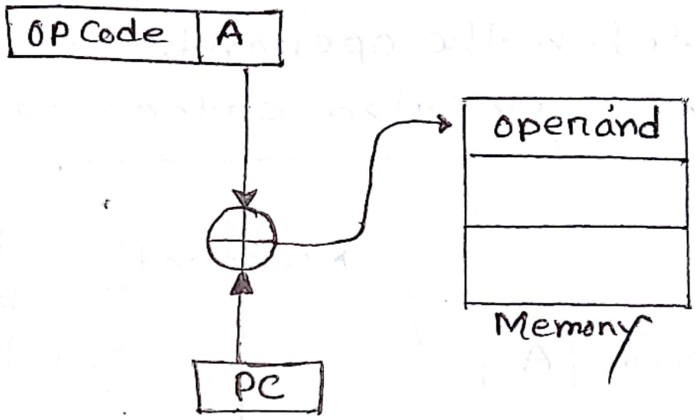
In this mode,

- Effective address of the operand is obtained by adding the content of program counter with the address part of the instruction.

Effective Address =

Content of program counter + Address part of the instruction

Note: program Counter (PC) always contains the address of the next instruction to be executed.



Relative Addressing Mode

by- Sorowar Mahabub

Q9) Subroutine Search -

```
MAIN PROGRAM
        MOVE #1000, R1      ; R1 ← 1000
        MOVE N, R0          ; R0 ← N
        CALL SUBROUTINE SEARCH

SUBROUTINE SEARCH
        CLEAR R3            ; R3 ← 0 This register is set to 1
                              when the value is found
        MOVE #1000, R1      ; R1 ← 1000
        MOVE N, R0          ; R0 ← N
Next:   COMPARE (R1)+ VAL   ; Use auto increment mode
        BRANCH_IF_EQUAL Found ; Branch if R1 equal to VAL
        DECREMENT R0        ; R0 ← R0 - 1
        BRANCH_IF>0 NEXT
        COMPARE R0, 0       ; Are all values in list checked
        BRANCH_IF_EQUAL, Not Found
Found:  MOVE #1, R3         ; Value VAL found at position R1-1
Not_Found:
        RETURN SEARCH
```

fig: Search Subroutine.

Prepared by: Sonowar Mahabub
C201032
Cell: 01521564157