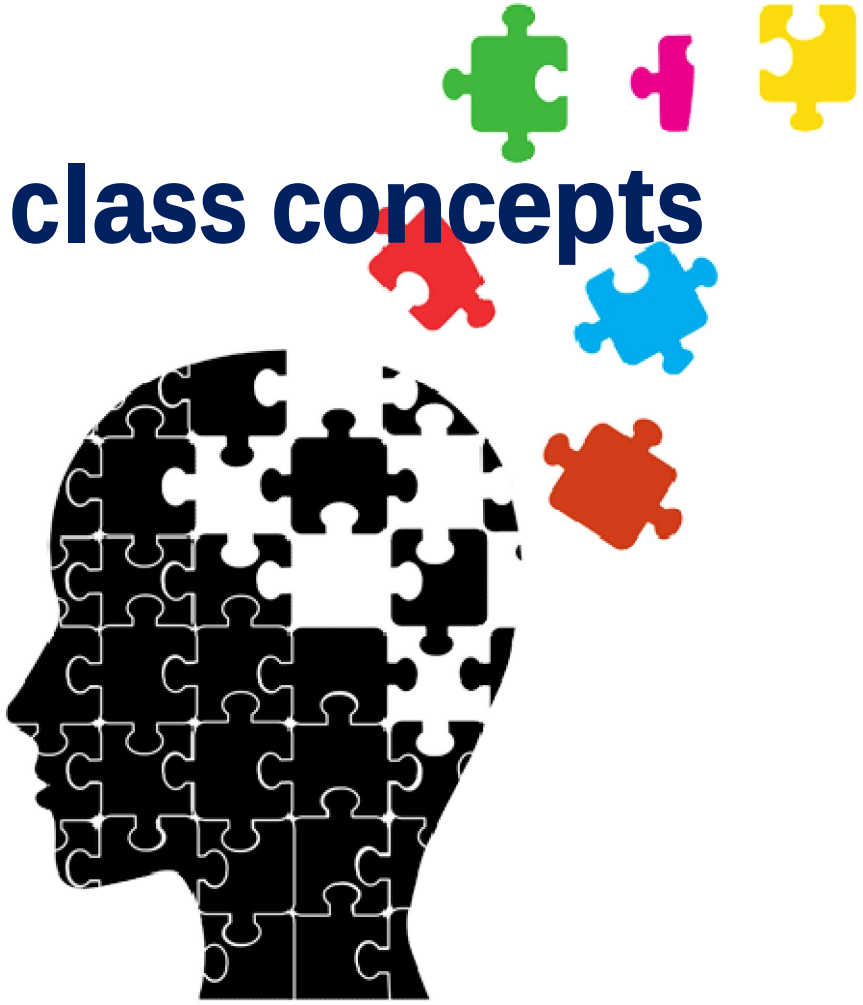
UNIT I
BASIC STRUCTURE OF COMPUTERS

Functional units – Basic operational concepts – Bus Structures – Performance – Memory locations and addresses – Memory operations – **Instruction and Instruction sequencing** – Addressing modes – Assembly language – Case study : RISC and CISC Architecture.

Recall the previous class concepts



Instruction and Instruction sequencing

- Sequence of small steps, such as
  - ✓adding two numbers,
  - ✓testing for a particular condition,
  - ✓reading a character from the keyboard, or
  - ✓ sending a character to be displayed on a display screen.

Types of operation

4 types

- Data transfers between the memory and the registers - (MOV, PUSH, POP, XCHG)
- Arithmetic and logic operations on data - (ADD, SUB, MUL, DIV,AND, OR, NOT).
- Program sequencing and control - (CALL.RET, LOOP, INT).
- I/0 transfers - (IN, OUT).

Register Transfer Notation (RTN)

Location	Hardware Binary Address	Example	Description
Memory	LOC, PLACE, NUM	R1 ← [LOC]	Contents of memory-location LOC are transferred into register R1.
Processor	R0, R1 ,R2	[R3] ← [R1]+[R2]	Add the contents of register R1 &R2 and places their sum into R3.
I/O Registers	DATAIN, DATAOUT	R1 ← DATAIN	Contents of I/O register DATAIN are transferred into register R1.

Assembly Language Notation

Assembly Language Format	Description
Move LOC, R1	Transfer data from memory-location LOC to register R1. The contents of LOC are unchanged by the execution of this instruction, but the old contents of register R1 are overwritten.
Add R1, R2, R3	Add the contents of registers R1 and R2, and places their sum into register R3.

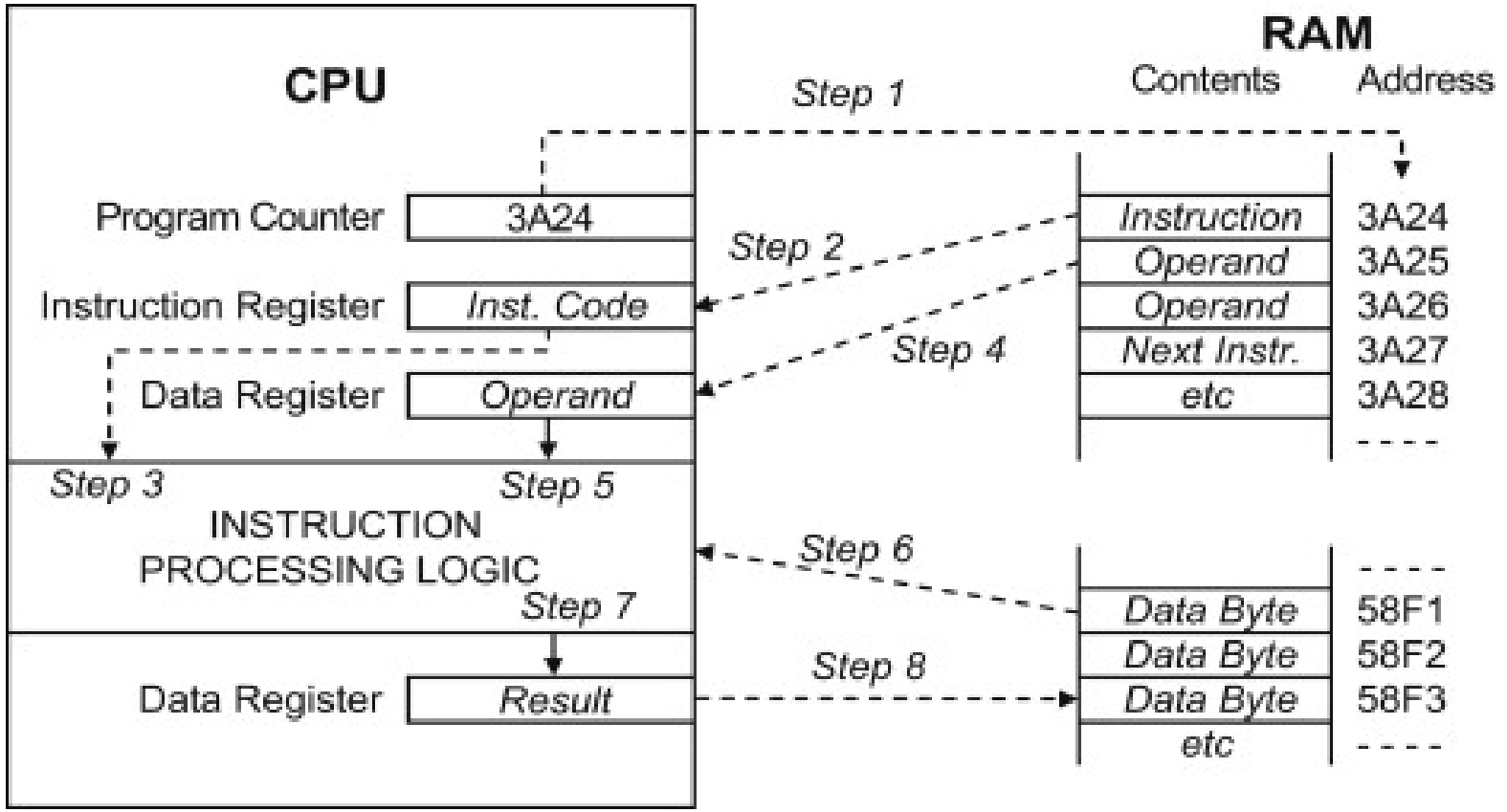
Basic Instruction Types

Instruction Type	Syntax	Example	Description	Instructions for Operation C<-[A]+[B]
Three Address	Opcode Source1,Source2,Destination	Add A,B,C	Add the contents of memory-locations A & B. Then, place the result into location C.	
Two Address	Opcode Source, Destination	Add A,B	Add the contents of memory-locations A & B. Then, place the result into location B, replacing the original contents of this location. Operand B is both a source and a destination.	Move B, C Add A, C
One Address	Opcode Source/Destination	Load A	Copy contents of memory-location A into accumulator.	Load A Add B Store C
		Add B	Add contents of memory-location B to contents of accumulator register & place sum back into accumulator.	
		Store C	Copy the contents of the accumulator into location C.	
Zero Address	Opcode [no Source/Destination]	Push	Locations of all operands are defined implicitly. The operands are stored in a pushdown stack.	Not possible

Instruction Execution & Straight Line Sequencing

- Initially, the address of the first instruction is loaded into PC
- Then, the processor control circuits use the information in the PC to fetch and execute instructions, one at a time, in the order of increasing addresses. This is called Straight-Line sequencing.
- During the execution of each instruction, PC is incremented by 4 to point to next instruction.

Instruction Execution & Straight Line Sequencing



Instruction Execution & Straight Line Sequencing

2 phases for Instruction Execution:

Fetch Phase: The instruction is fetched from the memory-location and placed in the IR.

Execute Phase: The contents of IR is examined to determine which operation is to be performed. The specified-operation is then performed by the processor

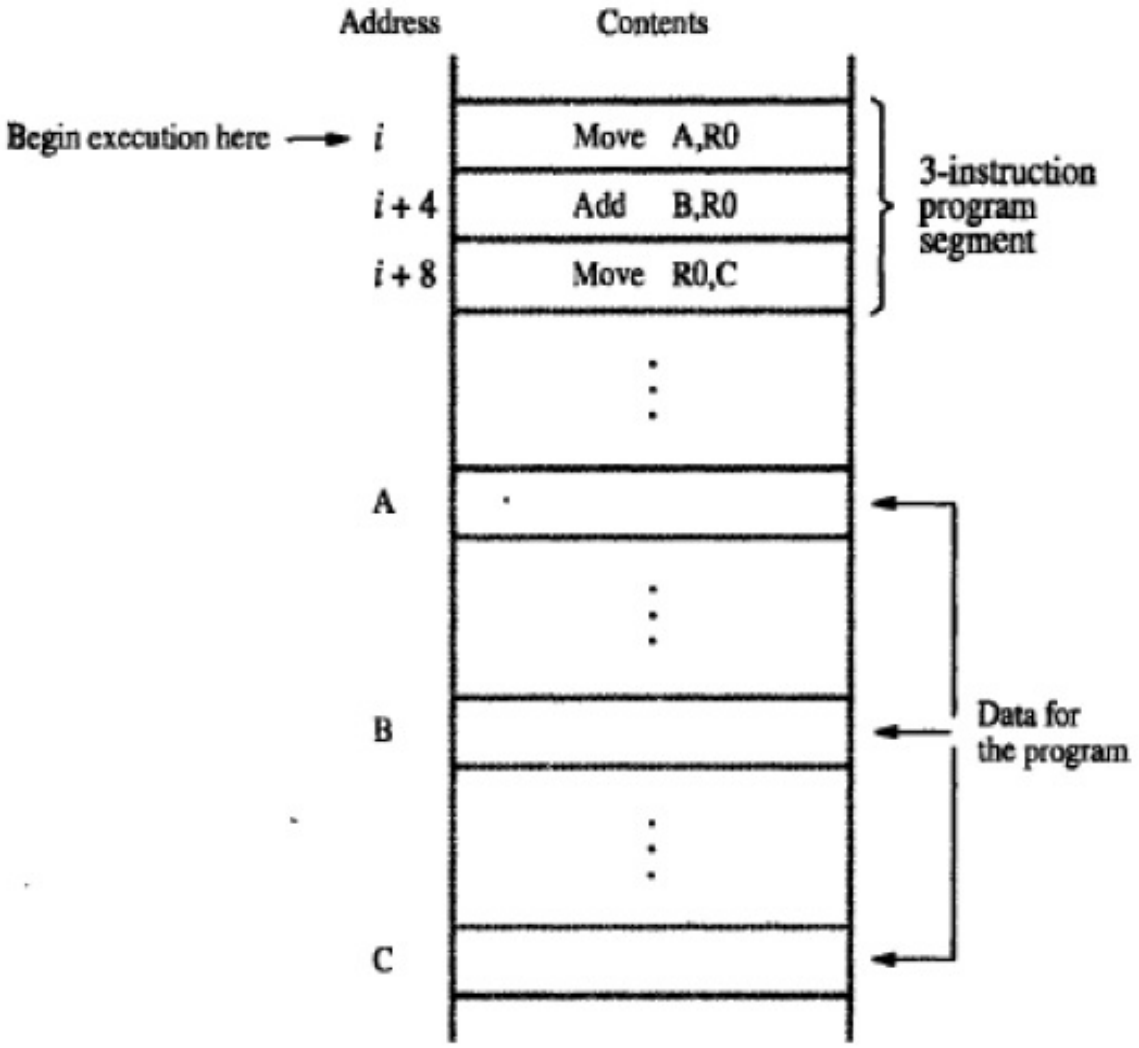


Figure 2.8 A program for C ← [A] + [B].

Branching

- Branch-type of instruction loads a new value into program counter.
- So processor fetches & executes instruction at this new address called "branch target"
- Conditional branch-causes a branch if a specified condition is satisfied.
- Example

  Branch>0 LOOP –conditional branch instruction .it executes only if it satisfies condition.

Condition codes

- Individual condition code flags-1 or 0.
- 4 commonly used flags.
  - N (negative)-set to 1 if result is –ve or else 0.
  - Z (zero)-set to 1 if result is 0, or else 0 .
  - V (overflow)-set to 1if arithmetic overflow occurs or else 0.
  - C(carry)-set to 1 if carry out results from operation or else 0

TEXT BOOK

Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, McGraw-Hill, 6th Edition 2012.

REFERENCES

1. David A. Patterson and John L. Hennessey, “Computer organization and design”, MorganKauffman ,Elsevier, 5th edition, 2014.
2. William Stallings, “Computer Organization and Architecture designing for Performance”, Pearson Education 8th Edition, 2010
3. John P.Hayes, “Computer Architecture and Organization”, McGraw Hill, 3rd Edition, 2002
4. M. Morris R. Mano “Computer System Architecture” 3rd Edition 2007
5. David A. Patterson “Computer Architecture: A Quantitative Approach”, Morgan Kaufmann; 5th edition 2011

THANK YOU