SNS COLLEGE OF ENGINEERING

Kurumbapalayam (PO), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE NAME: 19IT301 COMPUTER ORGANIZATION AND ARCHITECTURE

II YEAR/ III SEM

Unit 1 : BASIC STRUCTURE OF COMPUTERS Topic 7:

Assembly Language

Assembly Language

- Machine instructions are represented by patterns of 0s and 1s. So these patterns represented by symbolic names called "*mnemonics*"
- E.g. Load, Store, Add, Move, BR
- A complete set of such symbolic names and rules for their use constitutes a programming language, referred to as an *assembly language*.
- The set of rules for using the mnemonics in the specification of complete instructions and programs is called the *syntax* of the language.
  - Programs written in an assembly language can be automatically translated into a sequence of machine instructions by a program called an *assembler*.

Assembly Language

- The assembler program is one of a collection of utility programs that are a part of the system software of a computer.

- The user program in its original alphanumeric text format is called a **source program**, and the assembled machine-language program is called an **object program.**
- The assembly language for a given computer is not case sensitive

E.g.

- MOVE R1, SUM
- ADD #5,R3
- ADDI 5,R3

Assembler Directives

- In addition to representing instructions in a program, assembly language allows the programmer to specify other information needed to translate the source program into the object program.
- Assign numerical values to any names used in a program.
    - For e,g: name SUM is used to represent the value 20

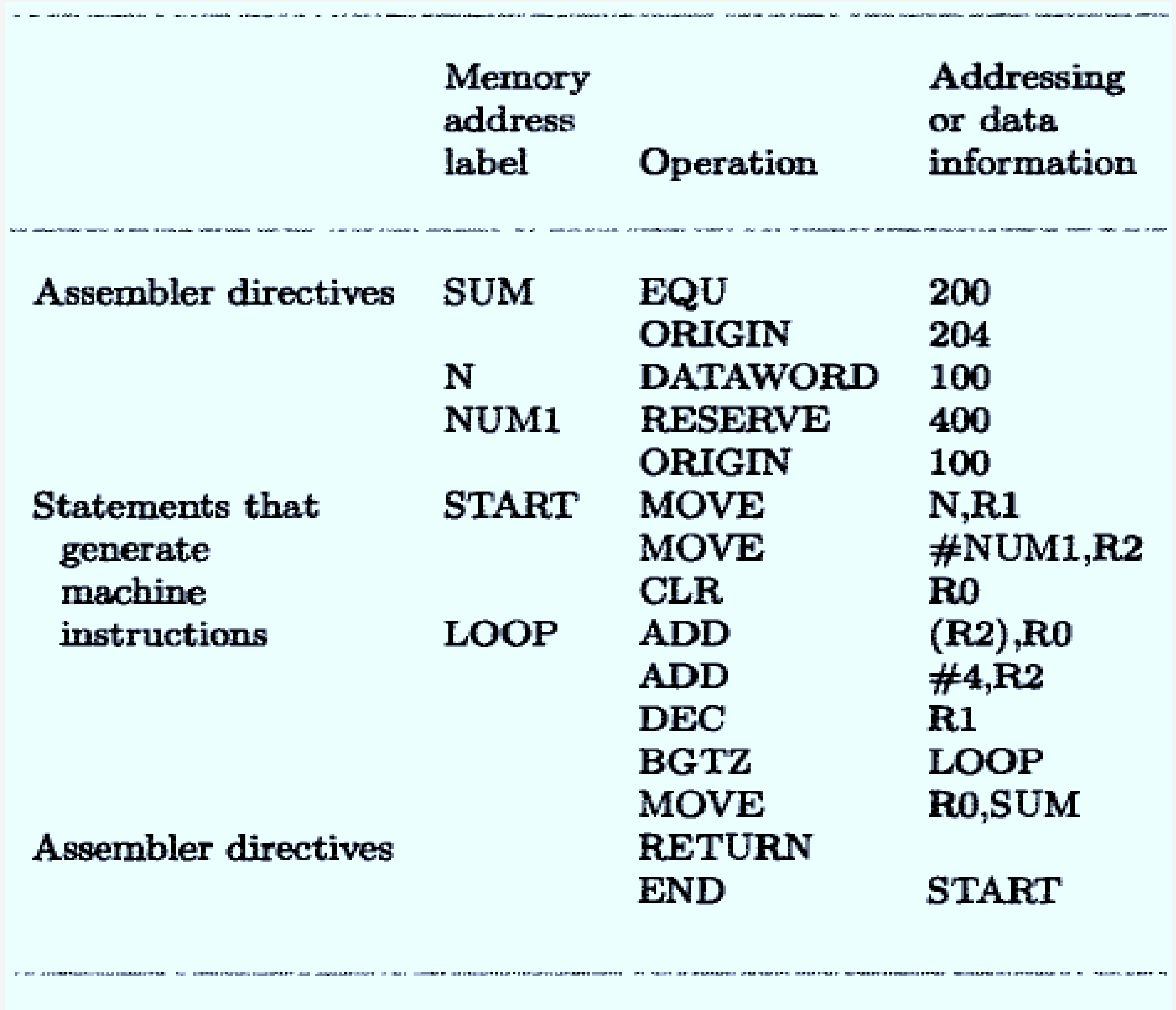
      SUM EQU 20 ;assembler directives(or commands)
- **Assembler directives** are instructions that direct the assembler to do something
    - Ex: EQU, ORIGIN, DS – Defines space.

Assembler Directives

- If the assembler is to produce an object program according to this arrangement, it has to know
  - How to interpret the names
  - Where to place the instructions in the memory
  - Where to place the data operands in the memory

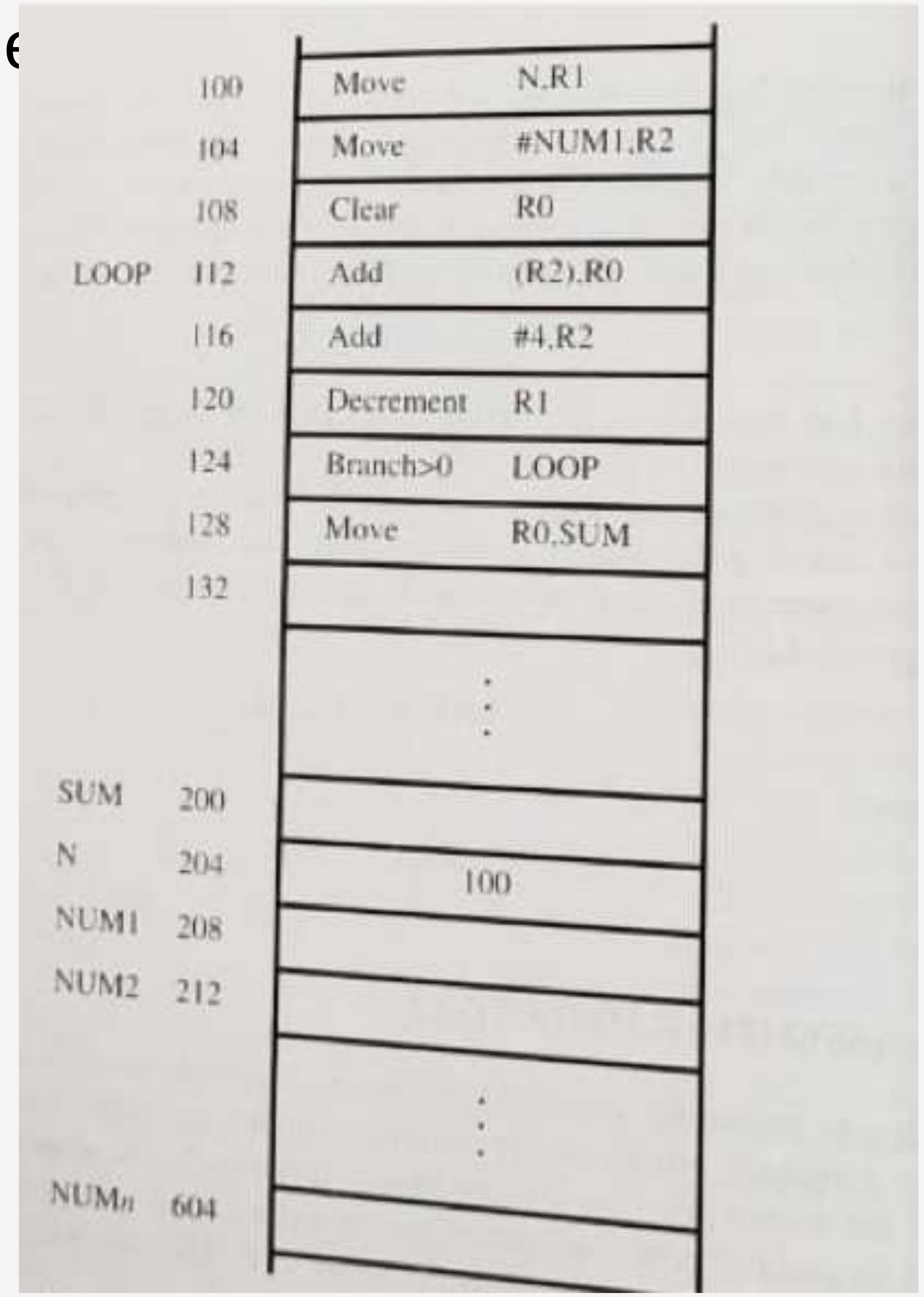
Assembly language representation for the program

Label: Operation Operand(s) Comme



	Memory address label	Operation	Addressing or data information
Assembler directives	SUM	EQU	200
		ORIGIN	204
	N	DATAWORD	100
	NUM1	RESERVE	400
		ORIGIN	100
Statements that generate machine instructions	START	MOVE	N,R1
		MOVE	#NUM1,R2
		CLR	R0
	LOOP	ADD	(R2),R0
		ADD	#4,R2
		DEC	R1
		BGTZ	LOOP
		MOVE	R0,SUM
Assembler directives		RETURN	
		END	START

Assembly language representation for the program



Memory Arrangement for addition of N numbers

General Format of a Statement

- Most assembly languages require statements in a source program to be written in the form:

Label	**Operation**	**Operands**	**Comment**

1) **Label** is an optional name associated with the memory-address
2) **Operation** field contains the OP-code mnemonic of the desired instruction or assembler.
3) Operand field contains addressing information for accessing one or more **operands**, depending on the type of instruction.

4) **Comment** field is used for documentation purposes to make program easier to understand

Assembly and Execution of Programs

- Programs written in an assembly language are automatically translated into a sequence of machine instructions by the Assembler.

Assembler Program

→ replaces all symbols denoting operations & addressing-modes with binary-codes used in machine instructions.

→ replaces all names and labels with their actual values.

→ assigns addresses to instructions & data blocks, starting at address given in ORIGIN directive

→ inserts constants that may be given in DATAWORD directives.

→ reserves memory-space as requested by RESERVE directives.

Two Pass Assembler

1) **First Pass:** Work out all the addresses of labels.

As the assembler scans through a source-program, it keeps track of all names of numerical- values that correspond to them in a **symbol-table.**

2) **Second Pass:** Generate machine code, substituting values for the labels.

When a name appears a second time in the source-program, it is replaced with its value from the table.

Assembly and Execution of Programs

- The assembler stores the object program on the secondary storage device available in the computer, usually a magnetic disk.
- An utility program **loader** loads the object program into the main memory
- **Debugger Program** is used to help the user find the programming errors
- Debugger program enables the user

  → to stop execution of the object-program at some points of interest &

  → to examine the contents of various processor-registers and memory-location.

Number Notation

- Decimal Number
  - ADD #93,R1
- Binary Number
  - ADD #%0101110,R1
- Hexadecimal Number
  - ADD #$5D,R1

Step 1: Divide (93)10 successively by 2 until the quotient is 0:
93/2 = 46, remainder is 1
46/2 = 23, remainder is 0
23/2 = 11, remainder is 1
11/2 = 5, remainder is 1
5/2 = 2, remainder is 1
2/2 = 1, remainder is 0
1/2 = 0, remainder is 1
Step 2: Read from the bottom (MSB) to top (LSB) as 1011101.

Assessment

1. __________ converts the programs written in assembly language into machine instructions.
a) Machine compiler
b) Interpreter
c) Assembler
d) Converter

2. The instructions like MOV or ADD are called as ______
a) OP-Code
b) Operators
c) Commands
d) None of the mentioned

Assessment

3. The purpose of the ORIGIN directive is __________
a) To indicate the starting position in memory, where the program block is to be stored
b) To indicate the starting of the computation code
c) To indicate the purpose of the code
d) To list the locations of all the registers used

4. _____ directive is used to specify and assign the memory required for the block of code.
a) Allocate
b) Assign
c) Set
d) Reserve

Answers

1. C
2. A
3. A
4. D

Thank You