ASSEMBLER PROJECT DOCUMENTATION

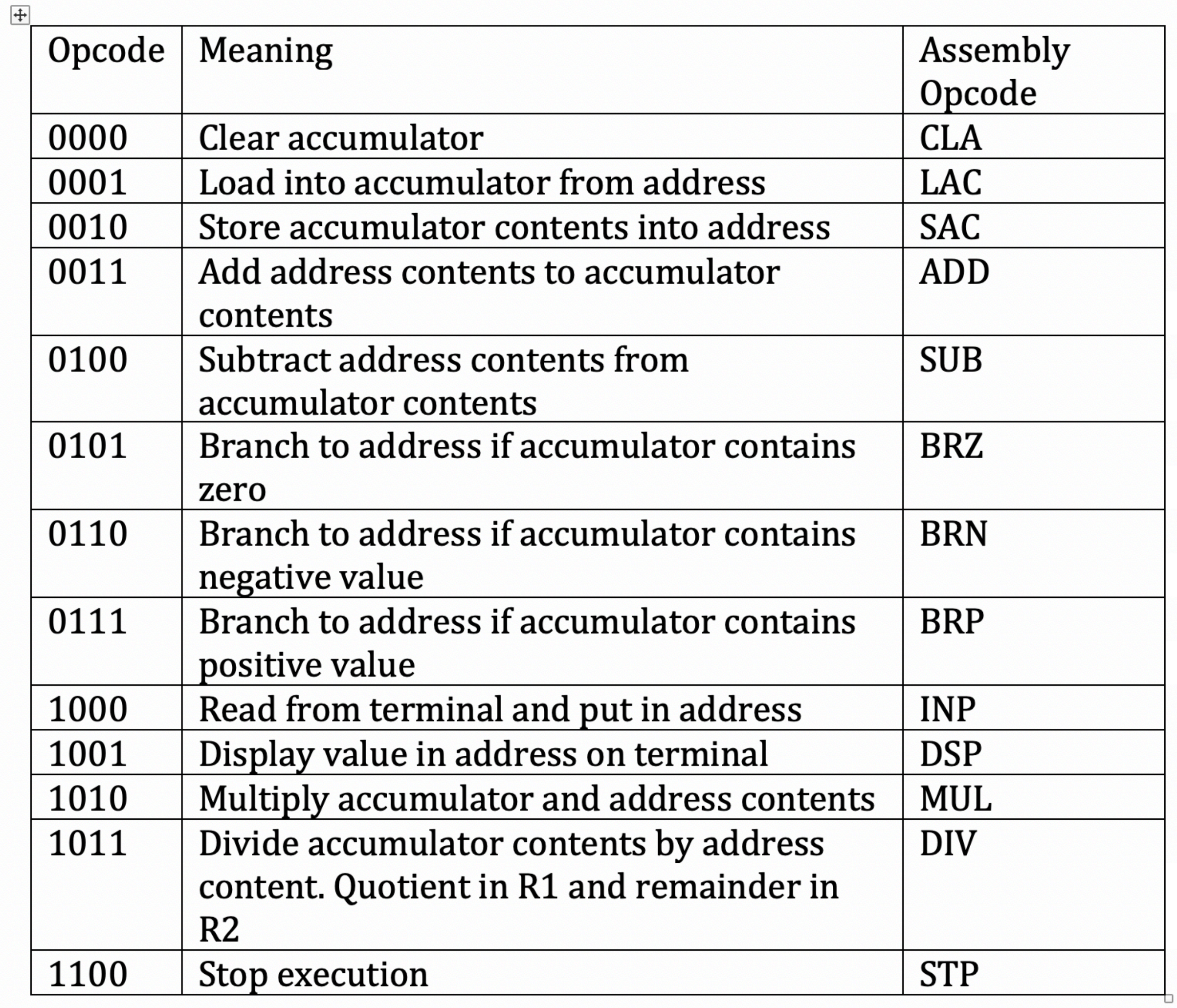
Anuneet Anand
Pankil Kalra

Design

Assembler is a program that converts assembly language code into machine language code. Our assembler is a 12 - bit assembler with first 4 bits reserved for Opcode and remaining 8 bits reserved for Memory.

Number Of Opcodes : 12
Memory Limit : 256

It is a two pass assembler.



Opcode	Meaning	Assembly Opcode
0000	Clear accumulator	CLA
0001	Load into accumulator from address	LAC
0010	Store accumulator contents into address	SAC
0011	Add address contents to accumulator contents	ADD
0100	Subtract address contents from accumulator contents	SUB
0101	Branch to address if accumulator contains zero	BRZ
0110	Branch to address if accumulator contains negative value	BRN
0111	Branch to address if accumulator contains positive value	BRP
1000	Read from terminal and put in address	INP
1001	Display value in address on terminal	DSP
1010	Multiply accumulator and address contents	MUL
1011	Divide accumulator contents by address content. Quotient in R1 and remainder in R2	DIV
1100	Stop execution	STP

Working

Reading :-

- It reads the Assembly code from a text file (“AssemblyCode.txt”).
- Empty lines are ignored while reading the file.
- Comments are parsed and collected separately.
- Location Counter is incremented for every line of code.
- Junk data is stored separately and appropriate error is shown

First Pass:-

- The assembly code is processed line by line.
- Opcode Table consisting of Opcodes & associated variables/labels is generated.
- Label Table consisting of Labels appearing as markers is generated.
- Symbol Table consisting of Variables & Labels appearing in branch statements is generated.
- Appropriate errors are checked.

Memory Allocation:-

- Memory is first allocated to instructions.
- It is followed by allocation of memory to variables.
- Appropriate checks are made for memory limit exceeded.

Second Pass:-

- Labels are resolved.
- Variables are substituted with their memory addresses.
- If no errors are present, Machine code is generated and the Tables are displayed on terminal.
- If errors are present, they are displayed with associated line numbers.

Writing:-

- The Machine code is written into a file (“MachineCode.txt”).

Syntax

+ Lines & Spaces:

- Empty lines & whitespaces are allowed in code.
- Tokens should be separated by spaces.

+Labels:

- It is mandatory for a label to end with ":" when appearing as a marker.
- Label can consist of any characters but it should start with an alphabet.

+Variables:

- Variable can consist of any characters but it should start with an alphabet.

+Opcodes:

- Opcodes should be written in capital letters and should be as given in further documentation.

+Comments:

- Use "//" to write a comment. Note that "//" can't appear in comment body or in code.
- Only single line comments are supported.
- Lines containing only comment is allowed.

Errors

- Memory Limit Exceeded
Note that memory size is limited to 255. This includes memory for both instructions and data. This error is shown when the number of instructions and variables exceed 255.

- Not Able To Resolve Stop
The assembly code should have exactly one stop and should appear at the end of the code. Any other occurrence of STP is treated as an error.

- Invalid Comment Declaration
This error is shown when comments are not declared as per syntax given.

- Invalid Symbols Detected/Received More Operands Than Expected
This error is shown when extra operands or junk data is given with an Opcode.

- Invalid Opcode/Label
The assembler has detected an invalid Opcode or Label.

- Multiple Opcodes Detected In One Instruction
One instruction of assembly code can have only one Opcode.

- Invalid Label/Variable Declaration
The variable or label was not declared as per the syntax provided.

- Missing Symbols/Received Less Operands Than Expected
The Opcode required an operand to work on but it was not supplied.

- Can't Resolve Label L
A branch instruction with L appeared in the code but no such Label was declared as a marker.

- Invalid Variable Clashes With A Label
A variable was given same name as label.