

Chapter 12: Assembly Language

WMC CS CLUB

April 28, 2022 (Year End)

§1 Introduction to Assembly

Programs written in high-level languages are traditionally converted by compilers into assembly language, which is turned into machine language programs – sequences of 1's and 0's – by an assembler. Even today, with very good quality compilers available, there is the need for programmers to understand assembly language. First, it provides programmers with a better understanding of the compiler and its constraints. Second, on occasion, programmers find themselves needing to program directly in assembly language in order to meet constraints in execution speed or space.

ACSL chose to define its own assembly language rather than use a “real” one in order to eliminate the many sticky details associated with real languages. The basic concepts of our ACSL topic description are common to all assembly languages.

§2 Reference Manual

Execution starts at the first line of the program and continues sequentially, except for branch instructions (BG, BE, BL, BU), until the end instruction (END) is encountered. The result of each operation is stored in a special word of memory, called the “accumulator” (ACC). The initial value of the ACC is 0. Each line of an assembly language program has the following fields:

LABEL OPCODE LOC

The LABEL field, if present, is an alphanumeric character string beginning in the first column. A label must begin with an alphabetic character (A through Z, or a through z), and labels are case-sensitive. Valid OPCODEs are listed in the chart below; they are also case-sensitive and uppercase. Opcodes are reserved words of the language and (the uppercase version) many not be used a label. The LOC field is either a reference to a label or immediate data. For example, “LOAD A” would put the contents referenced by the label “A” into the ACC; “LOAD =123” would store the value 123 in the ACC. Only those instructions that do not modify the LOC field can use the “immediate data” format. In the following chart, they are indicated by an asterisk in the first column.

OP CODE	DESCRIPTION
*LOAD	The contents of LOC are placed in the ACC. LOC is unchanged.
STORE	The contents of the ACC are placed in the LOC. ACC is unchanged.
*ADD	The contents of LOC are added to the contents of the ACC. The sum is stored in the ACC. LOC is unchanged. Addition is modulo 1,000,000.
*SUB	The contents of LOC are subtracted from the contents of the ACC. The difference is stored in the ACC. LOC is unchanged. Subtraction is modulo 1,000,000.
*MULT	The contents of LOC are multiplied by the contents of the ACC. The product is stored in the ACC. LOC is unchanged. Multiplication is modulo 1,000,000.
*DIV	The contents of LOC are divided into the contents of the ACC. The signed integer part of the quotient is stored in the ACC. LOC is unchanged.
BG	The contents of LOC are divided into the contents of the ACC. The signed integer part of the quotient is stored in the ACC. LOC is unchanged.
BE	Branch to the instruction labeled with LOC if $ACC = 0$.
BL	Branch to the instruction labeled with LOC if $ACC < 0$.
BU	Branch to the instruction labeled with LOC.
READ	Read a signed integer (modulo 1,000,000) into LOC.
PRINT	Print the contents of LOC.
DC	The value of the memory word defined by the LABEL field is defined to contain the specified constant. The LABEL field is mandatory for this opcode. The ACC is not modified.
END	Program terminates. LOC field is ignored and must be empty.

§3 Practice Problems

§3.1 Problems

- After the following program is executed, what value is in location TEMP?
- If the following program has an input value of N, what is the final value of X which is computed? Express X as an algebraic expression in terms of N.

TEMP	DC	0
A	DC	8
B	DC	-2
C	DC	3
	LOAD	B
	MULT	C
	ADD	A
	DIV	B
	SUB	A
	STORE	TEMP
	END	

	READ	X
	LOAD	X
TOP	SUB	=1
	BE	DONE
	STORE	A
	MULT	X
	STORE	X
	LOAD	A
	BU	TOP
DONE	END	

§3.2 Answer Key

1. The ACC takes on values -2, -6, 2, -1, and -9 in that order. The last value, -9, is stored in location TEMP.
2. This program loops between labels TOP and DONE for A times. A has an initial value of X and subsequent terms of N, then values of A-1, A-2, ..., 1. Each time through the loop, X is multiplied by the the current value of A. Thus, $X = A * (A-1) * (A-2) * \dots * 1$ or $X = A!$ or A factorial. For example, $5! = 5 * 4 * 3 * 2 * 1 = 120$. Since the initial value of A is the number input (i.e. N), the algebraic expression is $X = N!$.

Thank you for your involvement and support of WMC CS Club this year!
Your hard work and dedication will pay off in the future.
Best wishes to your future endeavours.