First Generation Languages 1GLs (Machine Language)

A machine language is an agreed-upon formalism, designed to code low-level programs as series of machine instructions. Using these instructions, the programmer can command the processor to perform arithmetic and logic operations, fetch and store values from and to the memory, move values from one register to another, test Boolean conditions, and so on. As opposed to high level languages, whose basic design goals are generality and power of expression, the goal of machine language's design is direct execution in, and total control of, a given hardware platform. Of course, generality, power, and elegance are still desired, but only to the extent that they adhere to the basic requirement of direct execution in hardware. Machine code, also known as machine language, is the elemental language of computers, comprising a long sequence of binary digital zeros and ones (bits).

Each processor has its own specific machine language, and it reads and handles a certain number of bits at a time. Because it is designed to know how many bits (and which bits) tell it what operation to do, the processor can look at the correct sequence of bits and perform the next operation. Then it reads the next instruction, and so on. Each machine-code instruction causes the CPU (central processing unit) to perform a simple operation such as an arithmetic calculation or storing data in RAM (random access memory). Execution of machine code can be controlled by firmware or else carried out by means of the CPU's internal wiring.

This chapter is language-oriented. Therefore, we can abstract away most of the details of the

underlying hardware platform. In particular, in order to give a general description of machine

languages, it is sufficient to focus on three main hardware elements only: a processor, a memory, and a set of registers

Machines: A machine language can be viewed as an agreed-upon formalism, designed to manipulate a memory using a processor and a set of registers.

Memory: The term "memory" refers loosely to the collection of hardware devices designed to store data and instructions. Some computer platforms store data and instructions in the same memory device, while others

employ different data and instruction memories, each featuring a separate address space. Conceptually speaking, all memories have the same structure: a continuous array of cells of some fixed width, also called words or locations, each having a unique address. Hence, an individual word (representing either a data item or an instruction) is specified by supplying its address. In what follows we will refer to such individual words using the notations Memory [address], RAM [address], or M[address] for brevity.

Processor: The processor, normally called Central Processing Unit or CPU, is a device capable of performing a fixed set of operations. These typically include arithmetic and logic operations, memory access operations, and control (also called branching) operations. The operands of these operations are the current values of registers and selected memory locations. Likewise, the results of the operations can be stored either in registers or in selected memory locations.

Registers: Memory access is a relatively slow operation requiring long instruction formats (an address may require 32 bits). For this reason, most processors are equipped with several registers, each capable of holding a single value. Located in the processor's immediate proximity, the registers serve as a high-speed local memory, allowing the processor to quickly store and retrieve data. This setting enables the programmer to minimize the use of memory access commands, thus speeding up the program's execution. In what follows we will refer to the registers as R0, R1, R2, etc.

A machine language program is a series of coded instructions. For example, a typical instruction in a 16-bit computer may be "1010001100011001". In order to figure out what this instruction means, we have to know the rules of the game, i.e. the instruction set of the underlying hardware platform. For example, the language may be such that each instruction consists of four 4-bit fields: the left-most field codes a CPU operation, and the remaining fields represent the operation's operands. Thus the above command may code the operation "set R3 to R1+R9", depending of course on the hardware specification and the machine language syntax.

For example adding the registers 1 and 2 and placing the result in register 6 is encoded:

[op	rs	rt	rd	shamt	funct]
0	1	2	6	0	32 decimal

000000 00001 00010 00110 00000 100000 binary

Load a value into register 8, taken from the memory cell 68 cells after the location listed in register 3:

[op | rs | rt | address/immediate]

35 3 8 68 decimal

100011 00011 01000 00000 00001 000100 binary

Jumping to the address 1024:

[op | target address]

2 1024 decimal

000010 00000 00000 00000 10000 000000 binary