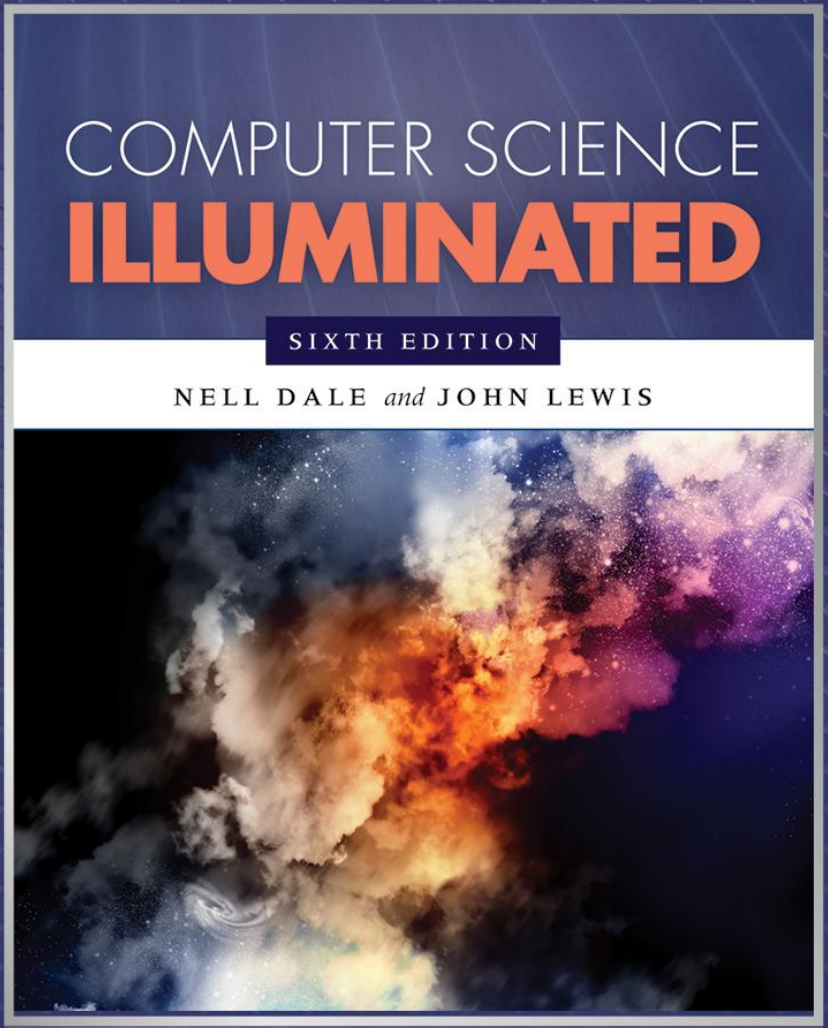
Chapter 6

Low-Level Programming Languages and Pseudocode



Chapter Goals

- Introduce machine languages at the binary level.
- Explore the Pep/8 virtual computer.
- Consider basic machine code for Pep/8.
- Write and run a small program in machine code.

Chapter Goals

- Describe the steps in creating and running an assembly-language program
- Write a simple program in assembly language
- Distinguish between instructions to the assembler and instructions to be translated
- Distinguish between following an algorithm and developing one
- Describe the pseudocode constructs used in expressing an algorithm

Computer Operations

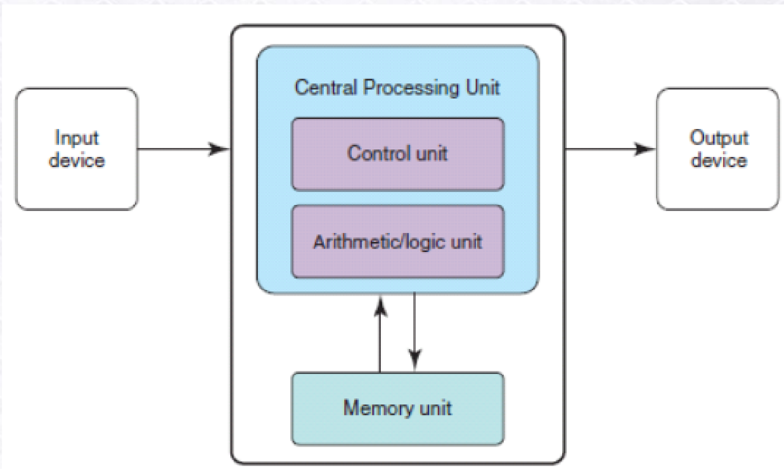
Computer

A programmable electronic device that can store, retrieve, and process data

Data and **instructions to manipulate** the data are logically the same and can be stored in the same place

What operations can a computer execute?

Computer Architecture (briefly)



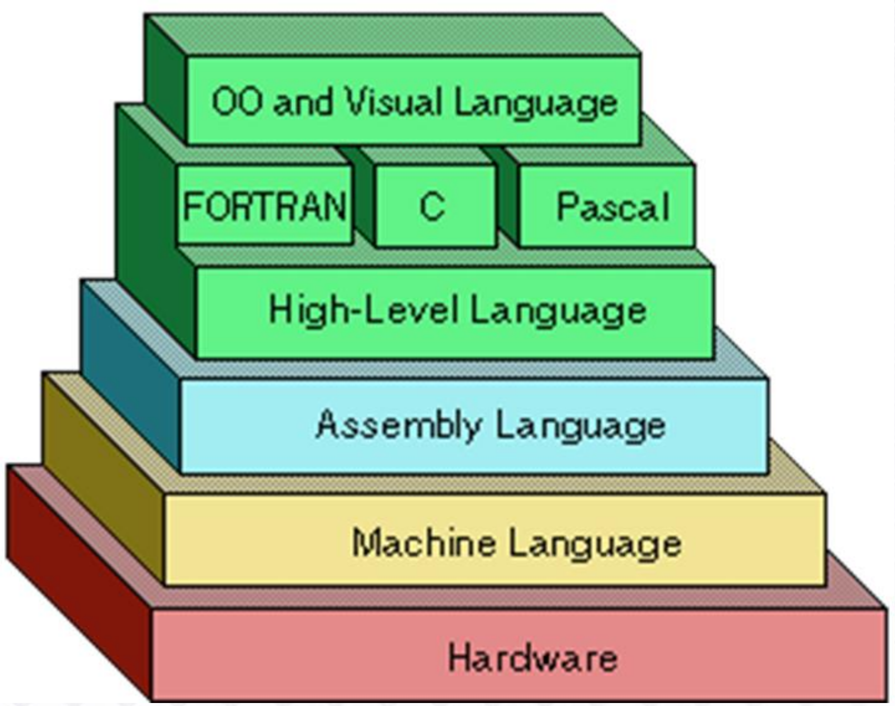
Address	Contents
00000000	11100011
00000001	10101001
⋮	⋮
11111100	00000000
11111101	11111111
11111110	10101010
11111111	00110011

Fetch

Execute

Decode

Language Abstraction



Machine Language

Machine language

The language made up of binary coded instructions built into the hardware of a particular computer and used directly by the computer

Why would anyone choose to use machine language?

(Hint: they had no choice. Why?)

Example Machine Code

```
01110011 01100101 01110010 01
01100101 01110010 00100000 01
01101000 01100001 01110100 00
01100100 01101001 01110011 01
01110010 01101001 01100010 01
01110100 01100101 01110011 00
01100001 01101110 01111001 00
01101001 01101110 01100011 01
01101101 01101001 01101110 01
00100000 01101101 01100101 01
01110011 01100001 01100111 01
01110011 00100000 01110100 01
00100000 01100001 01101100 01
00001101 00001010 00100000 00
```

(Disclaimer: I have no idea what this does)

Machine Language

Characteristics of machine language:

- Every processor type has its own specific set of machine instructions
- The digital logic of the CPU recognizes the binary representations of the instructions
- Each machine-language instruction does only one (typically) very low-level task

Pep/8 Virtual Computer

Virtual computer

A **hypothetical** machine designed to contain the important features of a real computer that we want to illustrate

Pep/8 http://computersystemsbook.com/4th-edition/pep8/

A virtual computer designed by Stanley Warford that has 39 machine-language instructions

Features in Pep/8

Pep/8 Registers/Status Bits

- The program counter ("PC") (contains the address of the next instruction to be executed)
- The instruction register ("IR") (contains a copy of the instruction being executed)
- The accumulator ("A") (used to hold data and results of operations)

The main memory unit is made up of 64KB (65,636 bytes) of storage

Architecture of Pep/8

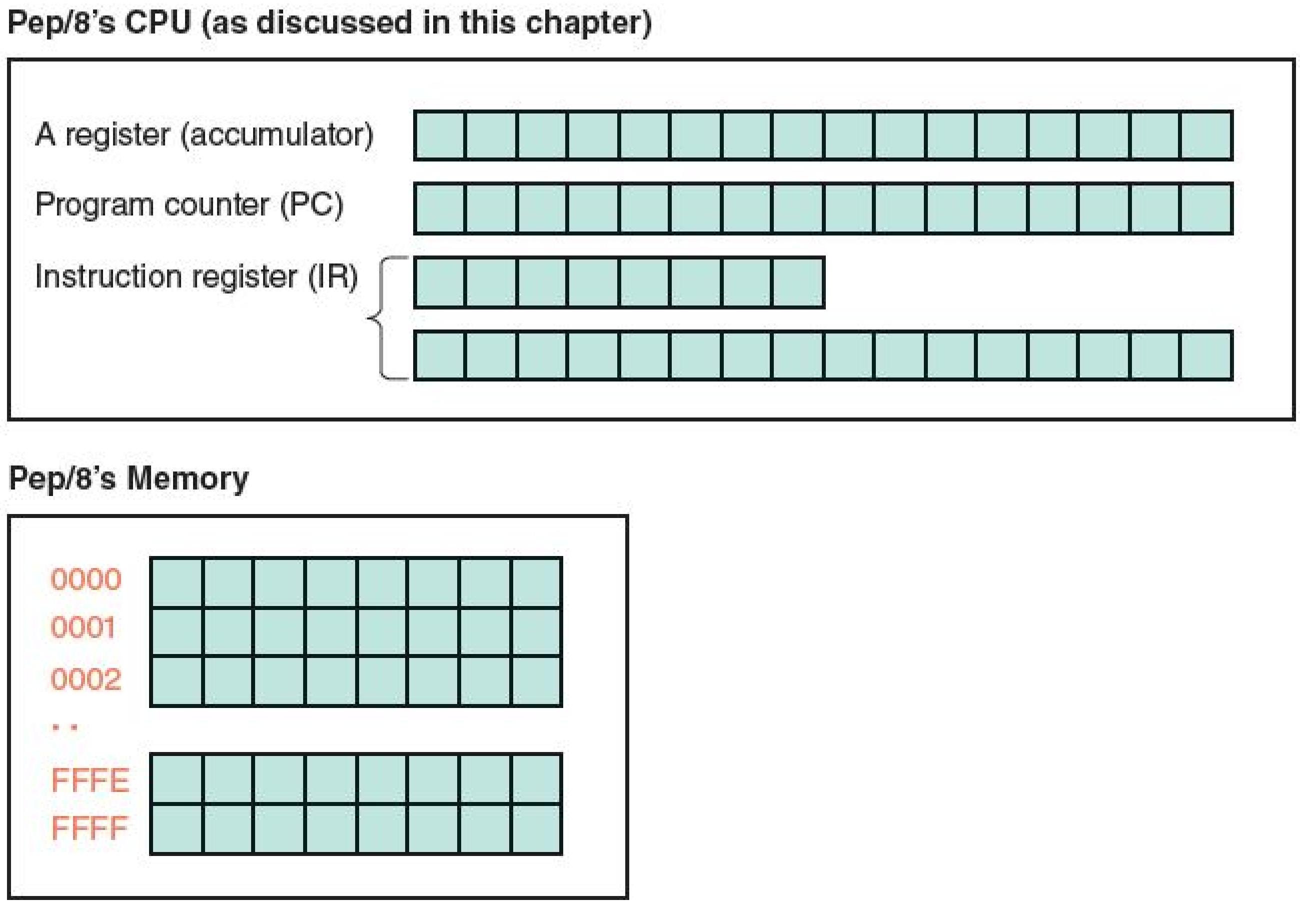


FIGURE 6.1 Pep/8's architecture

Instruction Format

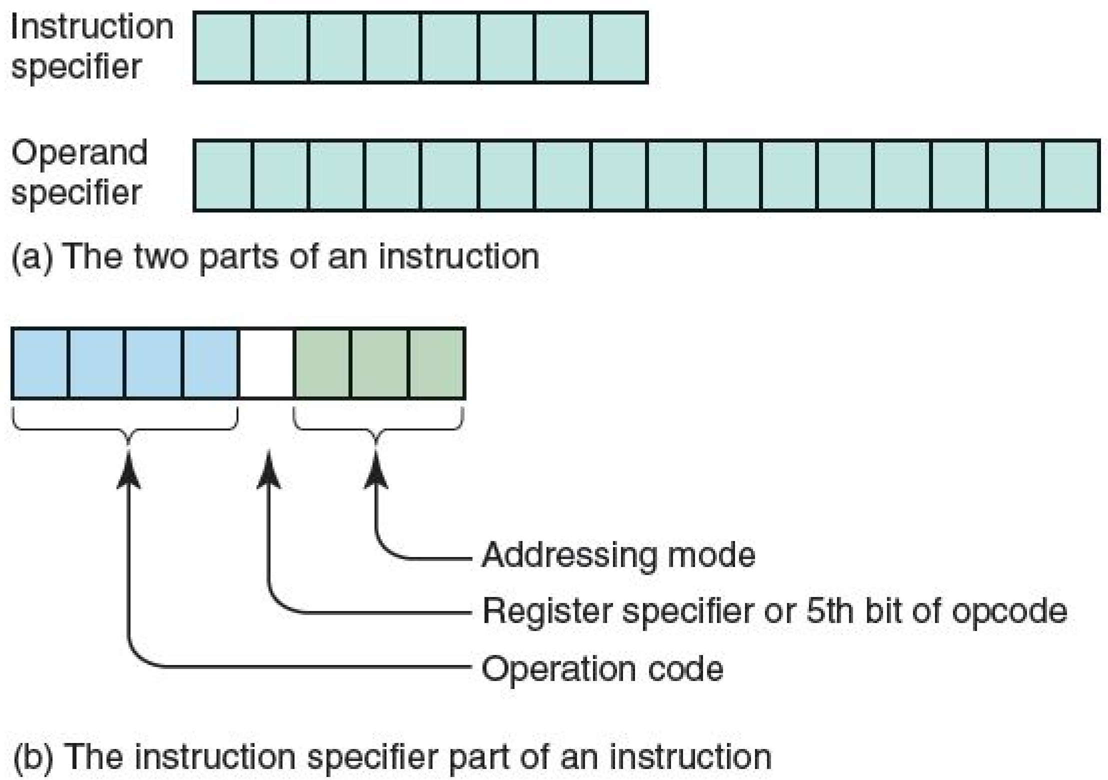
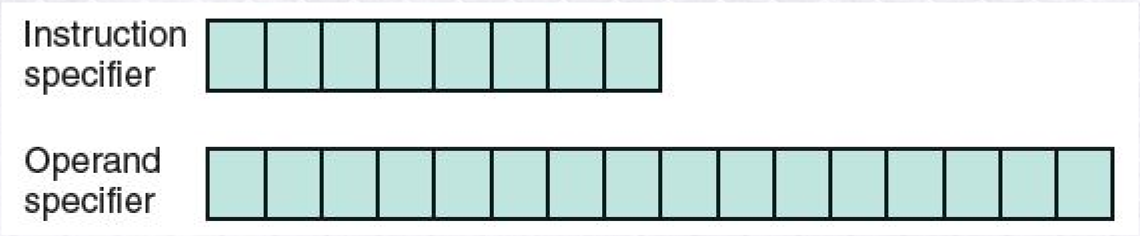


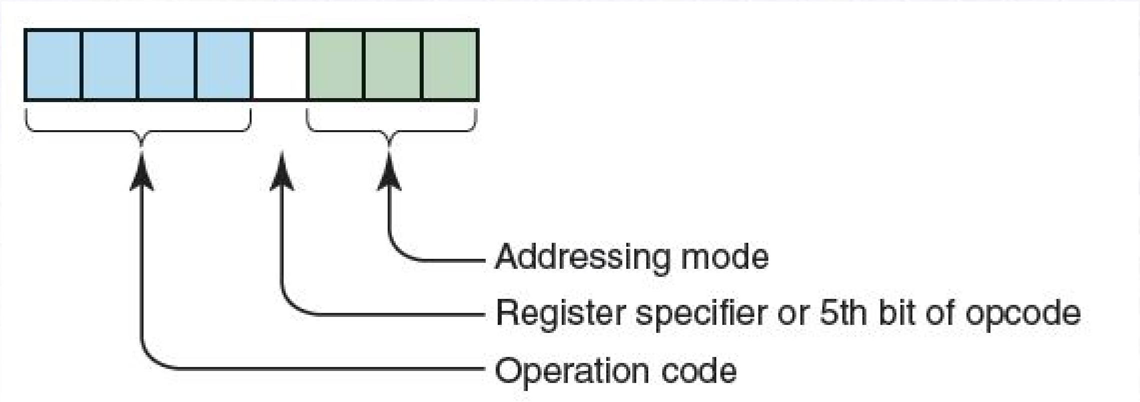
FIGURE 6.2 Pep/8 instruction format

Instruction Format

Each instruction is (optionally) made up of two parts:



The Instruction specifier is also divided as follows:



Instruction Format

Operation code

Specifies which instruction is to be carried out

Register specifier

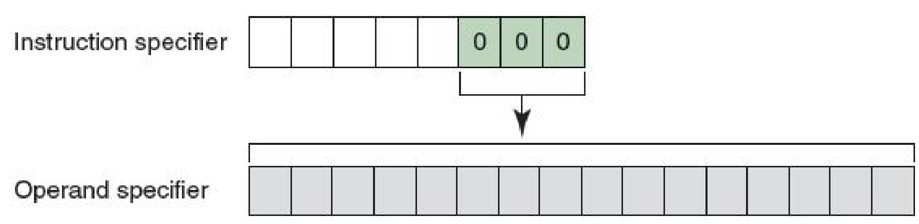
Specifies which register is to be used (for our purposes it always specifies the accumulator with a value of 0)

Addressing-mode specifier

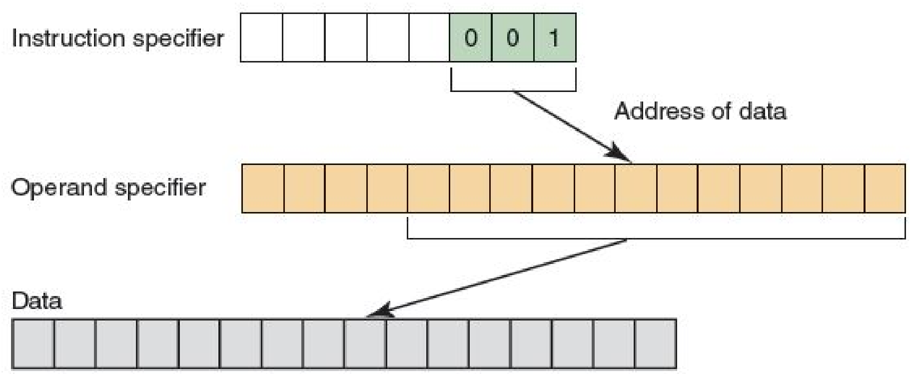
Says how to interpret the operand part of the instruction

Instruction Format

Addressing-mode
Allows us to say whether a value is stored in the operand specifier or in memory



(a) Immediate addressing mode: Operand is shaded gray



(b) Direct addressing mode: Operand is shaded gray

Instruction Format

Is there something we are not telling you about the addressing mode specifier? How can you tell?

Some Sample Instructions

Opcode	Meaning of Instruction
0000	Stop execution
1100	Load the operand into the A register
1110	Store the contents of the A register into the operand
0111	Add the operand to the A register
1000	Subtract the operand from the A register
01001	Character input to the operand
01010	Character output from the operand

FIGURE 6.4 Subset of Pep/8 instructions

Sample Instructions

What does the following do?

Instruction Specifier:

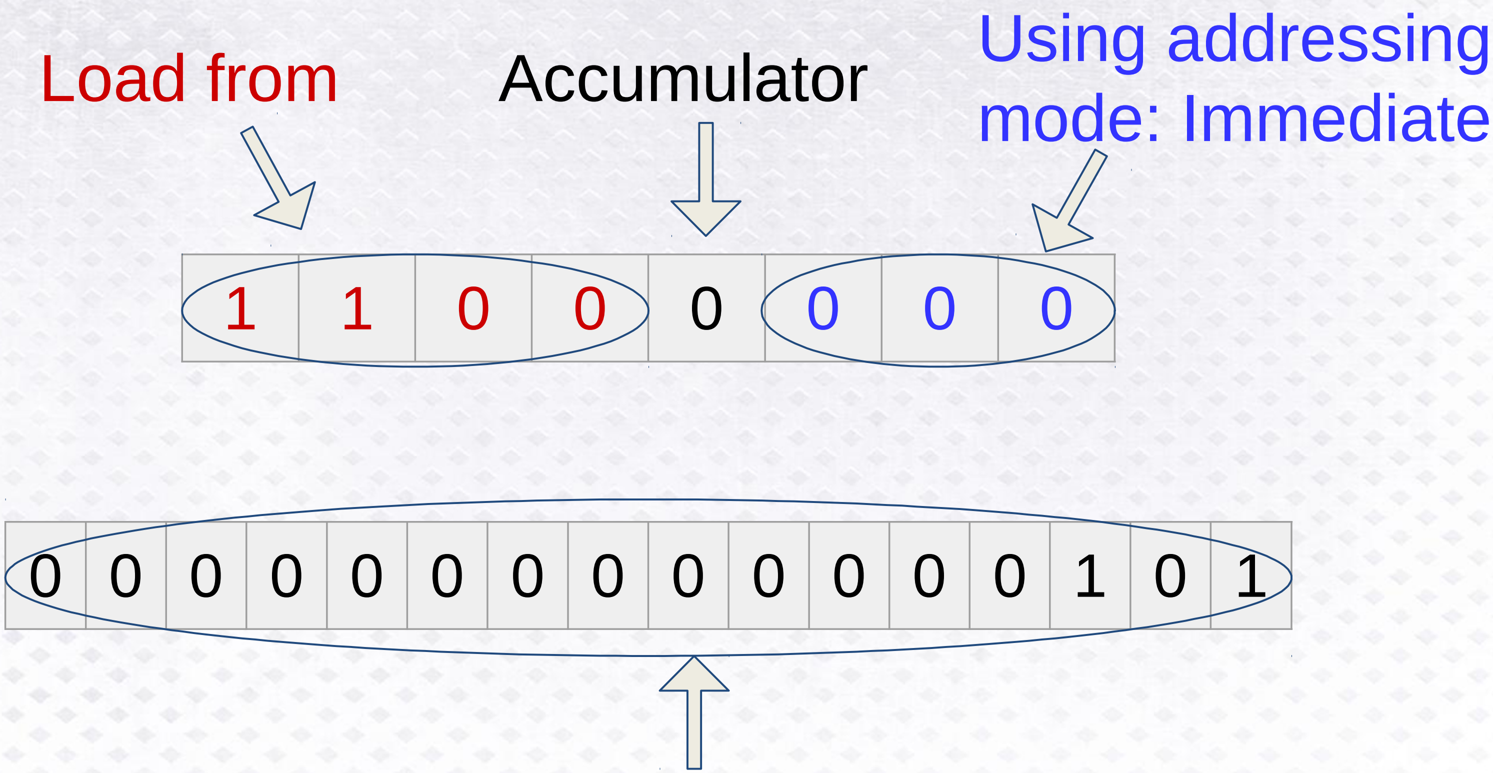
1	1	0	0	0	0	0	0

Operand Specifier:

0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

This loads the binary number 101 (i.e. 5) into the accumulator.

Sample Instructions



Because the addressing mode is immediate, this if the binary value loaded into the accumulator, i.e. decimal 5

Sample Instructions

What does the following do?

Instruction Specifier:

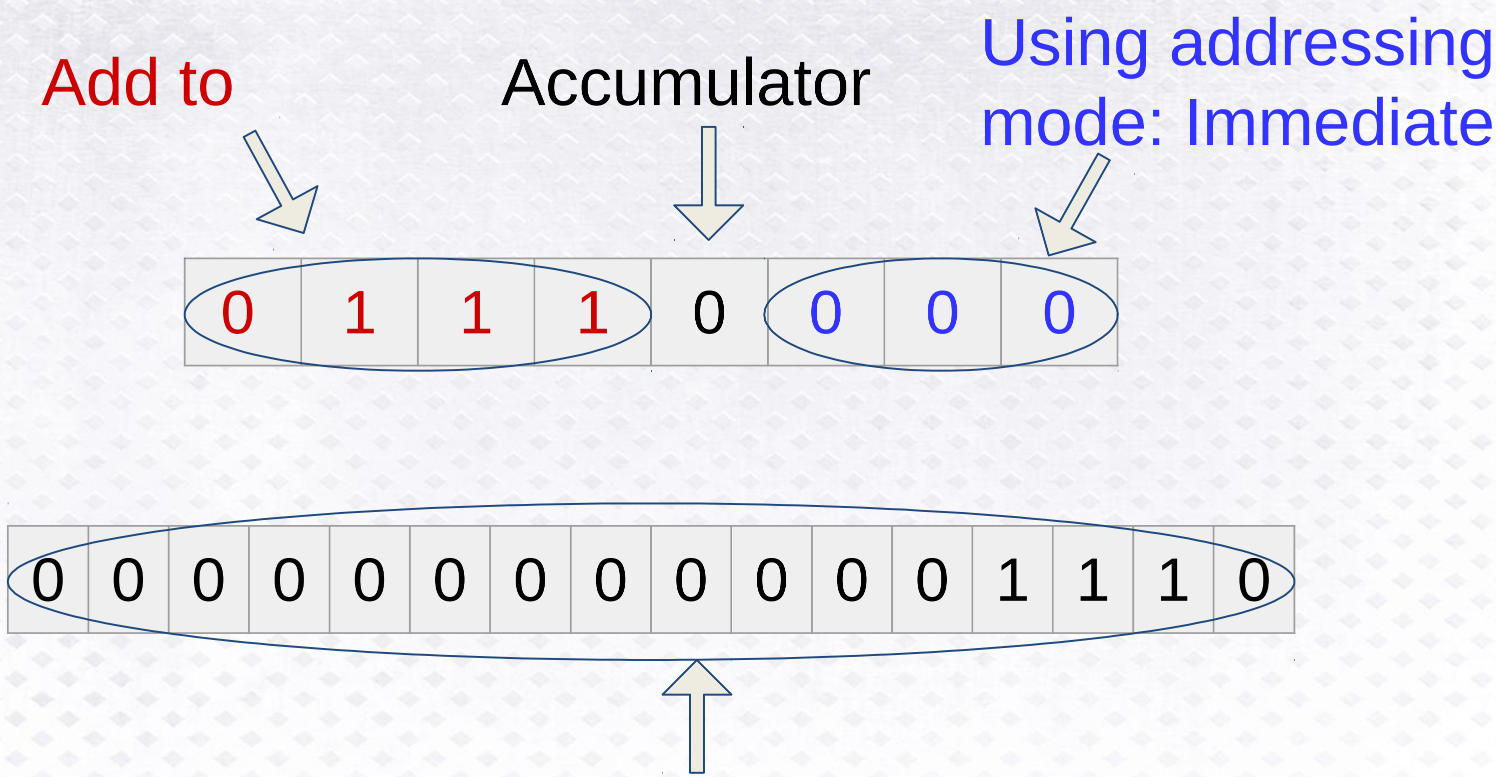
0	1	1	1	0	0	0	0

Operand Specifier:

0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

This adds the binary number 1110 (i.e. 14) into the accumulator.

Sample Instructions



Because the addressing mode is immediate, this if the binary value added into the accumulator, i.e. decimal 14

Sample Instructions

What does the following do?

Instruction Specifier:

1	0	0	0	0	0	0	0

Operand Specifier:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

This subtracts the binary number 11 (i.e. 3) from the accumulator.

Sample Instructions

Subtract from Accumulator Using addressing mode: Immediate

1	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Because the addressing mode is immediate, this if the binary value subtracted from the accumulator, i.e. decimal 3

Sample Instructions

What does the following do?

Instruction Specifier:

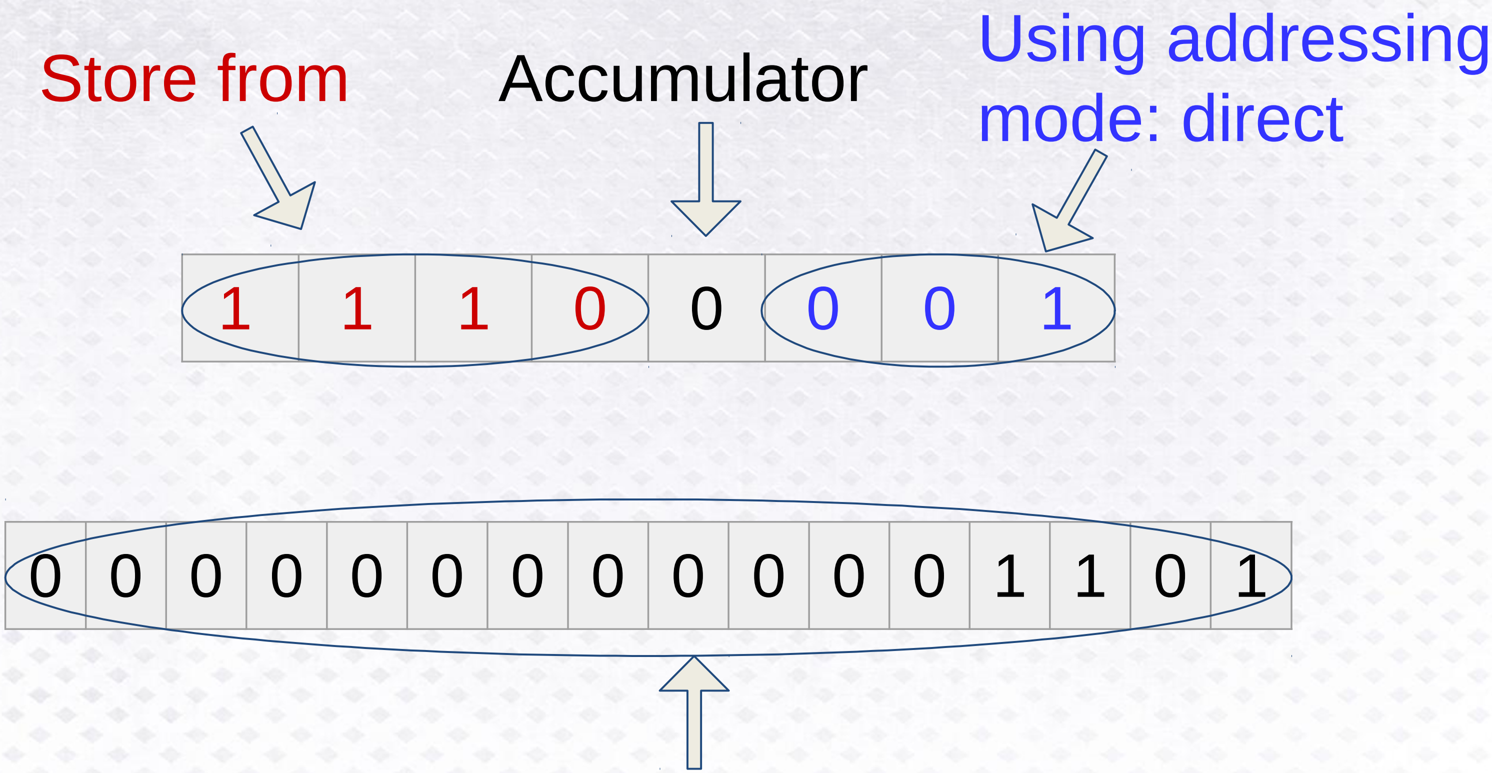
1	1	1	0	0	0	0	1

Operand Specifier:

0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

This stores the value from the accumulator into the memory address 1101.

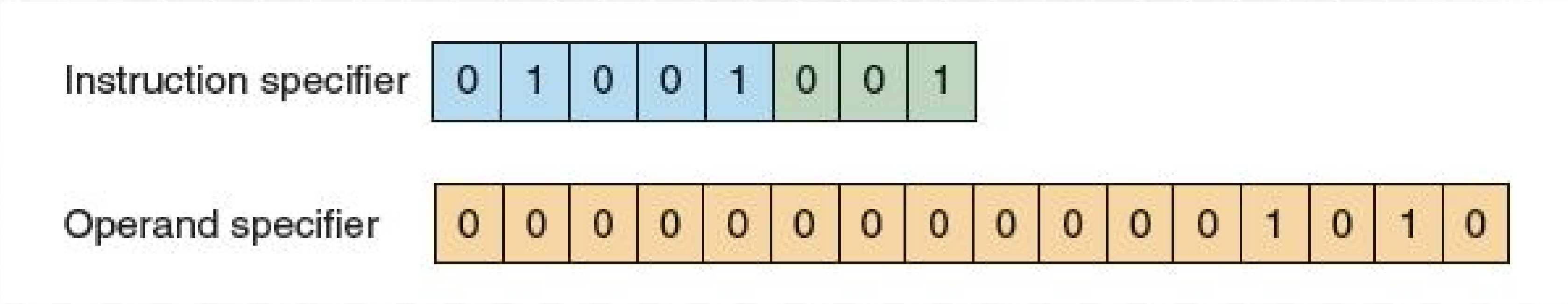
Sample Instructions



Because the addressing mode is direct, this if the binary value in the operand specifier is used as the location for storage.

Sample Instructions

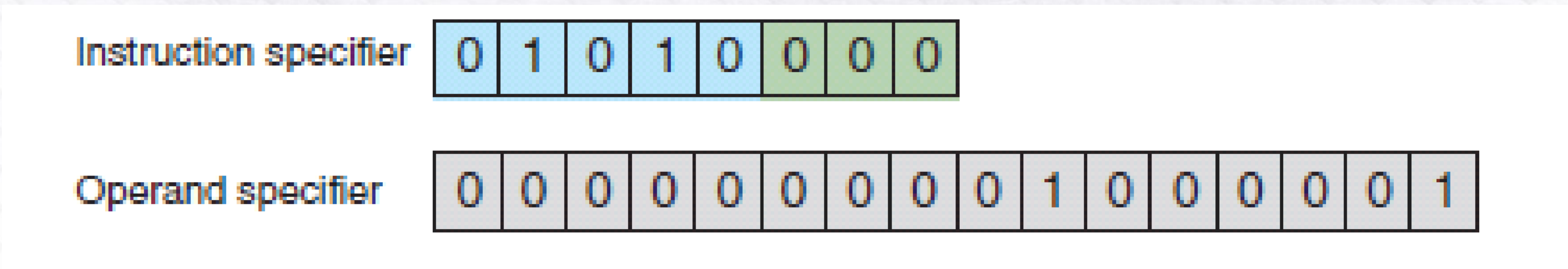
What do these instructions mean?

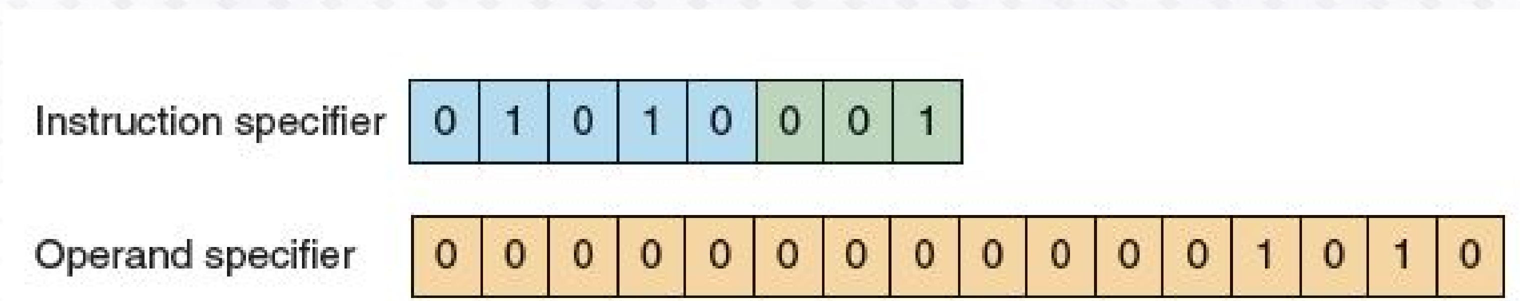


Why is there only one on this page?

Sample Instructions

What do these instructions mean?

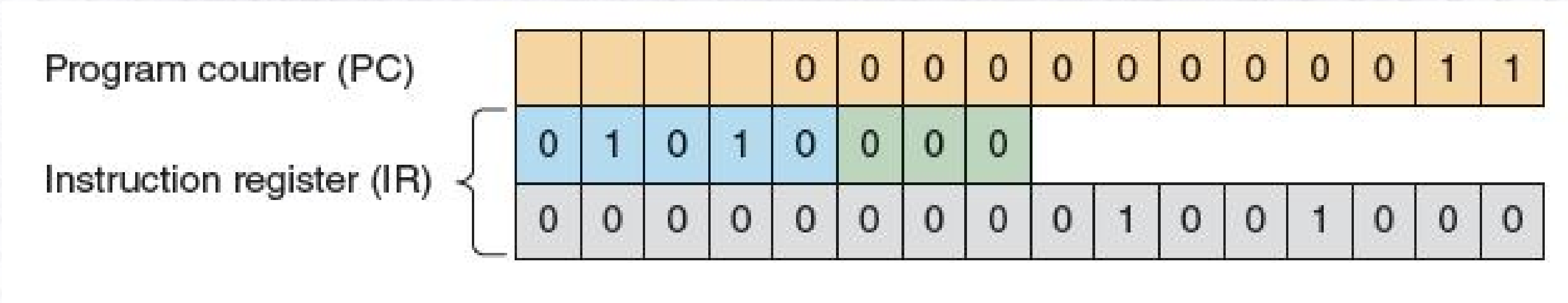




Written Algorithm of Hello

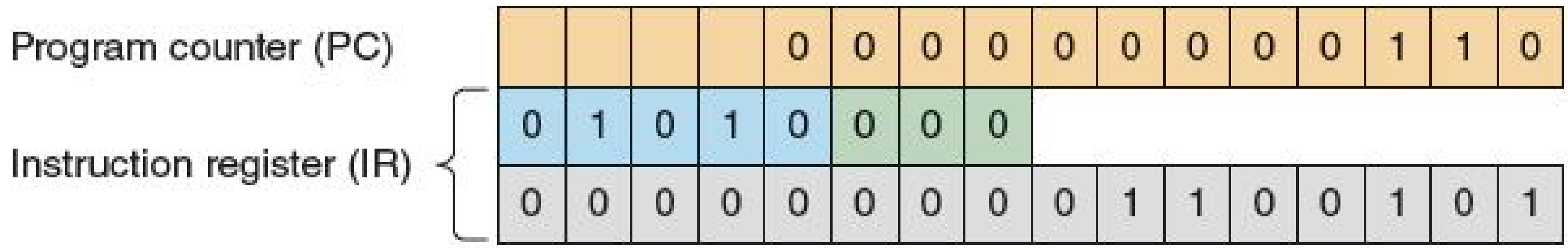
Action	Binary Instruction	Hex Instruction
Write ‘H’	01010000 0000000001001000	50 0048
Write ‘e’	01010000 0000000001100101	50 0065
Write ‘l’	01010000 0000000001101100	50 006C
Write ‘l’	01010000 0000000001101100	50 006C
Write ‘o’	01010000 0000000001101111	50 006F
Stop	00000000	00

Hand Simulation



Where in the fetch/execute cycle is this?
How much is the PC incremented?

Hand Simulation



Where in the fetch/execute cycle is this?

Example Program

Let’s try to add the numbers 4 and 5 together and store the result in memory address 00001100.

What would we need to do?

Steps required:

1. Load the number 5 into the accumulator.
2. Add the number 4 to the accumulator.
3. Store the answer (the value held in the accumulator into memory location 00001100).

Example Program

Load the number 5 into the accumulator:

1	1	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Add the number 4 to the accumulator:

0	1	1	1	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Store the result:

1	1	1	0	0	0	0	1

0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Example Program

Full Program:

11000000000000000000000010
10111000000000000000000001
00111000010000000000000001
10000000000000000000000000
0000