16.070

Introduction to Computers & Programming

Computer Architecture, Machine Language, Program Execution

Prof. Kristina Lundqvist
Dept. of Aero/Astro, MIT

Chapter Summary

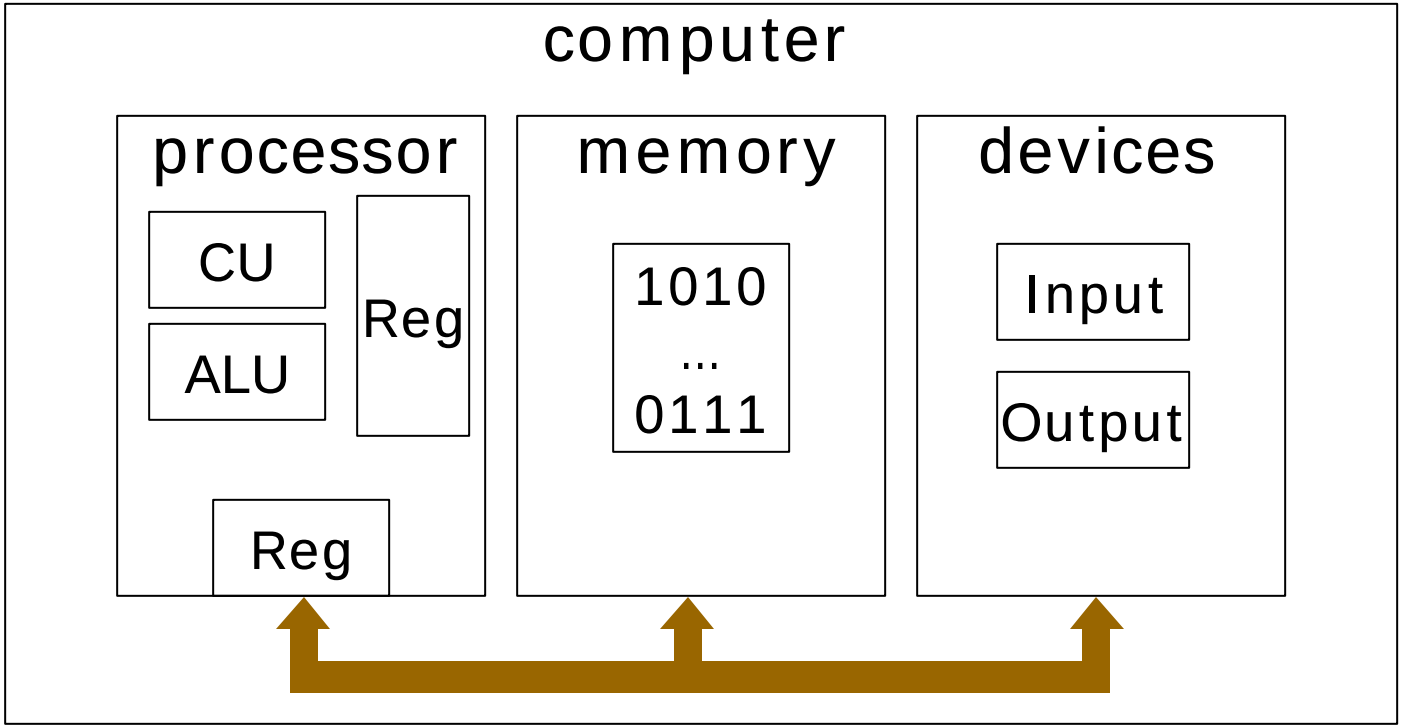
- This chapter introduces the activities of a computer's CPU. It describes the machine cycle executed by the control unit and the various operations (or, and, exclusive or, add, shift, etc.) performed by a typical arithmetic/logic unit. The concept of a machine language is presented in terms of the simple yet representative machine described in *Appendix C* of the text.

Computer Architecture

Computer architecture =

Interface between Computer and User =

instruction set architecture + computer organization



Computer Organization

- CPU: central processing unit
  - Part of a computer that controls all the other parts
  - Control Unit: fetches instructions from memory, decodes them and produces signals which control the other parts of the computer.
  - Arithmetic Logic Unit: performs operations such as addition, subtraction, bit-wise AND, OR, …
  - Memory:
    - Registers, Cache, RAM/ROM
  - Temporary buffers
  - Logic

Adding values stored in memory

Example 1

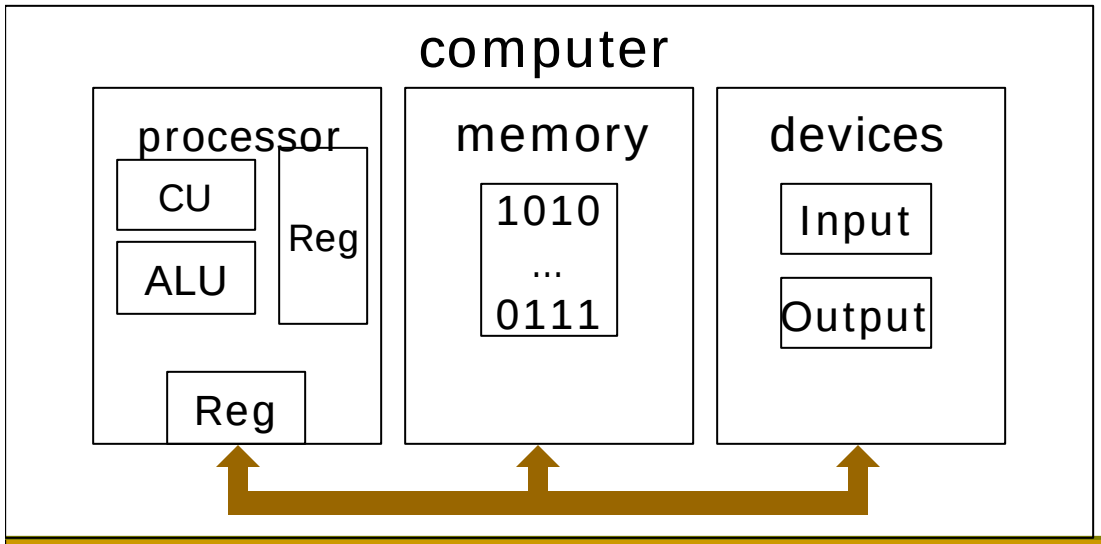
Step 1. Get one of the values to be added from memory and place it in a register.

Step 2. Get the other value to be added from memory and place it in another register.

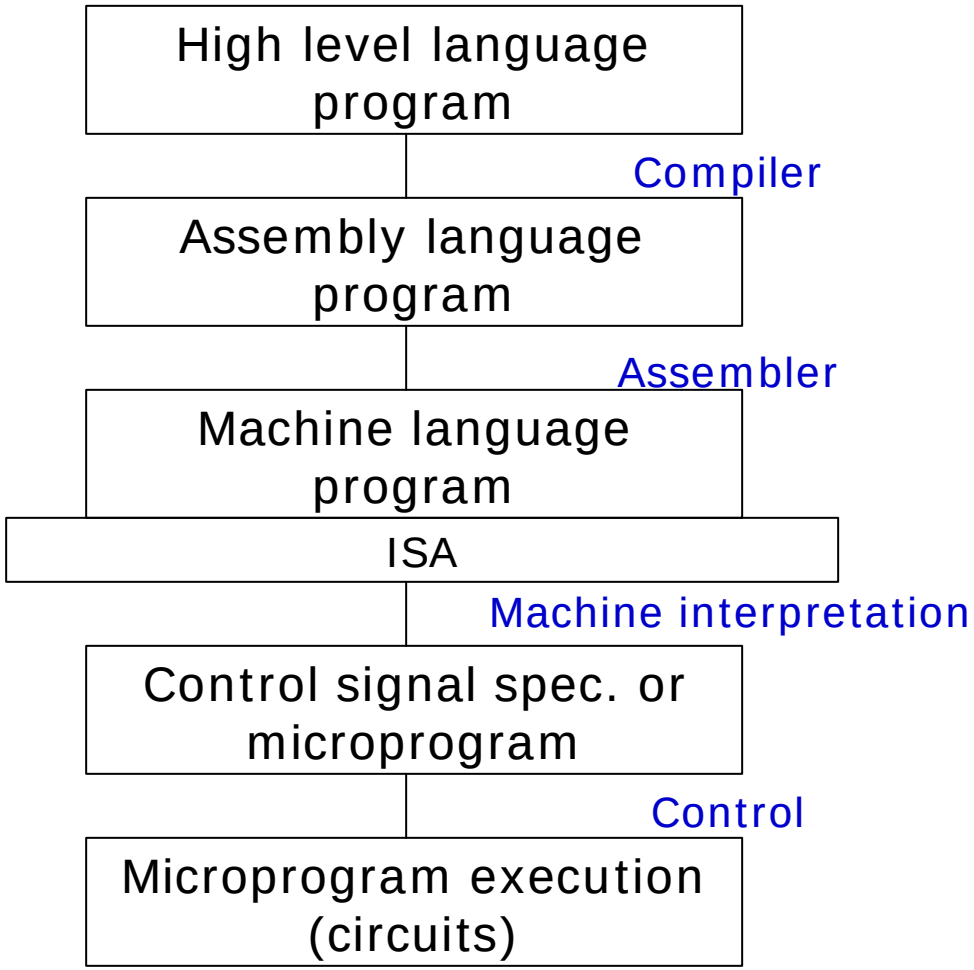
Step 3. Activate the addition circuitry with the registers used in Steps 1 and 2 as inputs and another register designated to hold the result.

Step 4. Store the result in memory.

Step 5. Stop.



Levels of Abstraction and Representation




```
Pay := Amount1 + Amount2;
Account := Account - Pay;
```

```
LOAD    R1,($2)
LOAD    R2,($4)
ADD     R2,R1,R2
STORE   R2,($6)
```

```
0000 0011 0011 0100 1101 1110
1001 0001 1011 1101 1100 0100
```

```
PCout, ARin, READ,
DRout, R1in, PCinc
ARin, READ
```

```
ALU operation
```

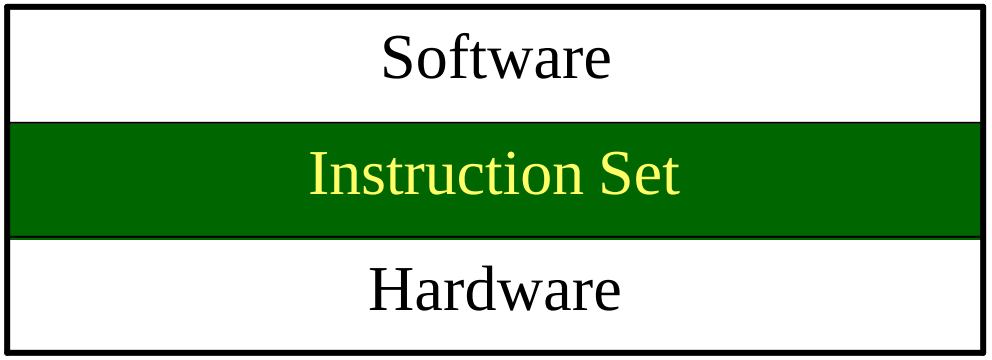
Instruction Set Architecture

- ISA: the parts of a processor's design that needs to be understood in order to write assembly language
  - Operations (add, sub, mult, …, how is it specified)
  - Number of operands (0, 1, 2, 3)
  - Operand storage (where besides memory)
  - Memory address (how is memory location specified)
  - Type and size of operands (byte, int, float, …)
  - Other aspects
    - Parallelism
    - Encoding
    - Successor instruction
    - Conditions

Basic ISA Classes

- Accumulator
- Stack
- General purpose register
- Load/store

Instruction Set



- The collection of machine language instructions that a processor understands
- Instructions are bits with well defined fields
- Instruction format
  - A mapping from instruction to binary values
  - Which bit positions correspond to which parts of the instruction

Instruction Set

- RISC (reduced instruction set computer)
  - A processor whose design is based on the rapid execution of a sequence of simple instructions
- CISC (complex instruction set computer)
  - Each instruction can perform several low-level operations such as memory access, arithmetic operations or address calculations

The instruction repertoire

- Data transfer
  - LOAD / STORE
  - I/O instructions
- Arithmetic/logic
  - Instructions that tell the CU to request an activity within the ALU (+, -, …, XOR, SHIFT, ROTATE)
- Control
  - Instructions that direct the execution of the program
    - Conditional jumps
    - Unconditional jumps

Dividing values stored in memory

Step 1. LOAD a register with a value from memory.

Step 2. LOAD another register with another value from memory.

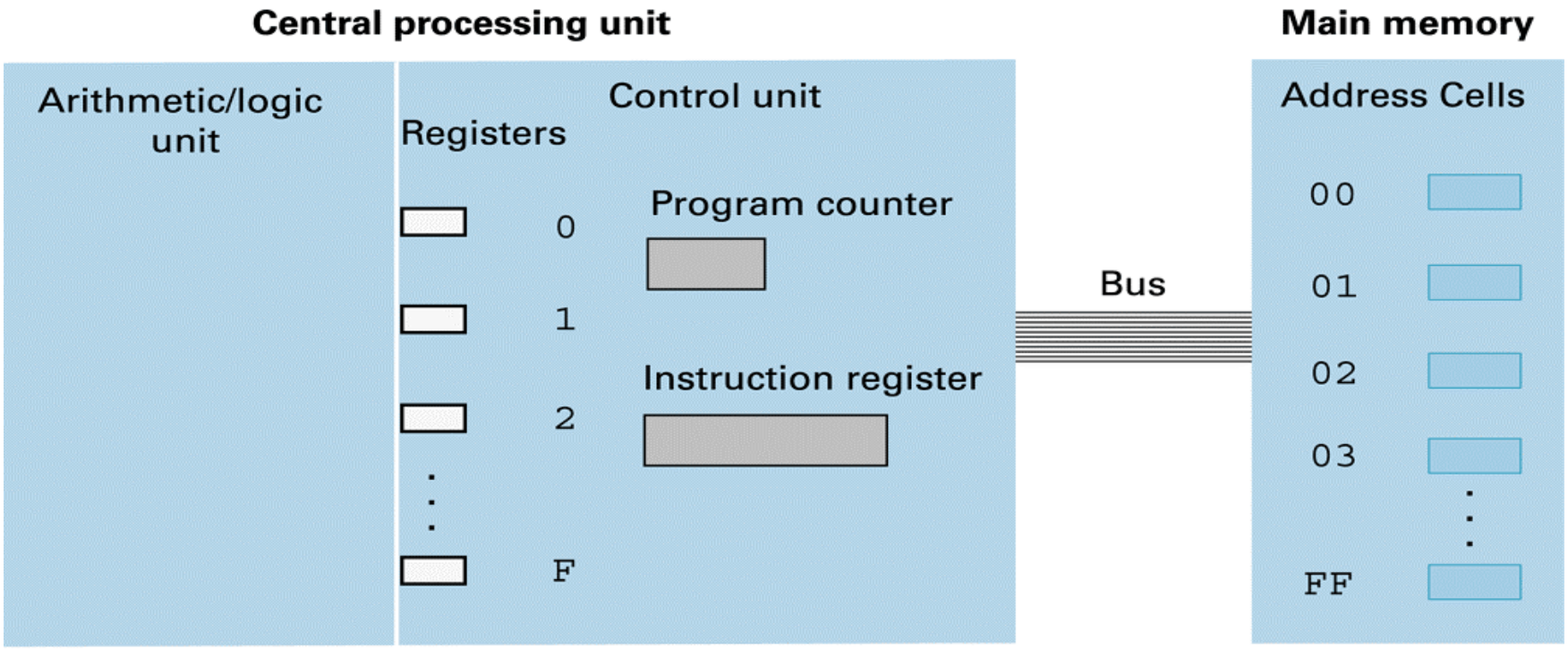
Step 3. If this second value is zero, JUMP to Step 6.

Step 4. Divide the contents of the first register by the second register and leave the result in a third register.

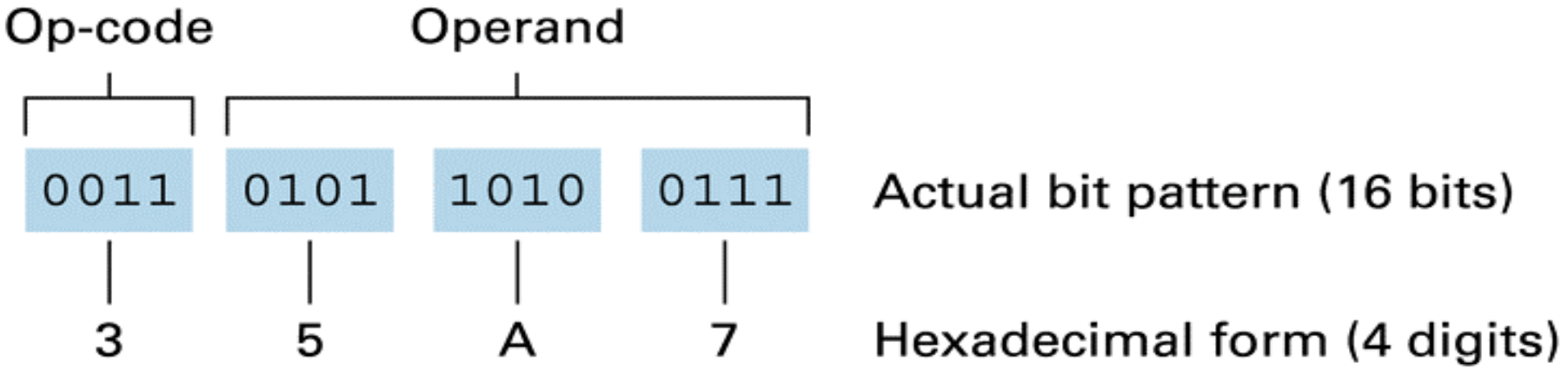
Step 5. STORE the contents of the third register in memory.

Step 6. STOP.

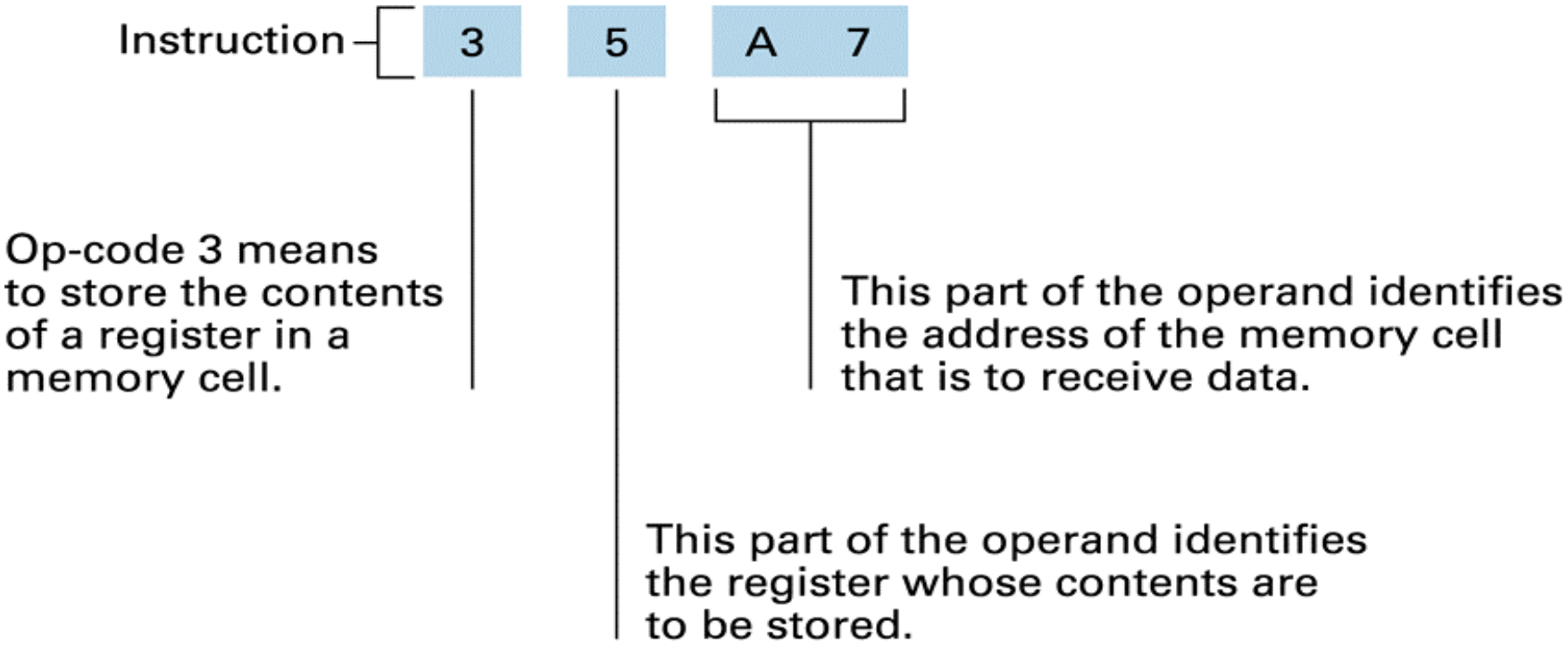
The architecture of the machine described in Appendix C



The composition of an instruction for the machine in Appendix C



Decoding the instruction 35A7



An encoded version of the instructions in Example 1

Example 2

Encoded instructions	Translation
156C	Load register 5 with the bit pattern found in the memory cell at address 6C.
166D	Load register 6 with the bit pattern found in the memory cell at address 6D.
5056	Add the contents of register 5 and 6 as though they were two's complement representation and leave the result in register 0.
306E	Store the contents of register 0 in the memory cell at address 6E.
C000	Halt.

Example in Ada

```
X : Integer := 92;
Y : Integer := 90;
Z : Integer;


Z := X + Y;
```

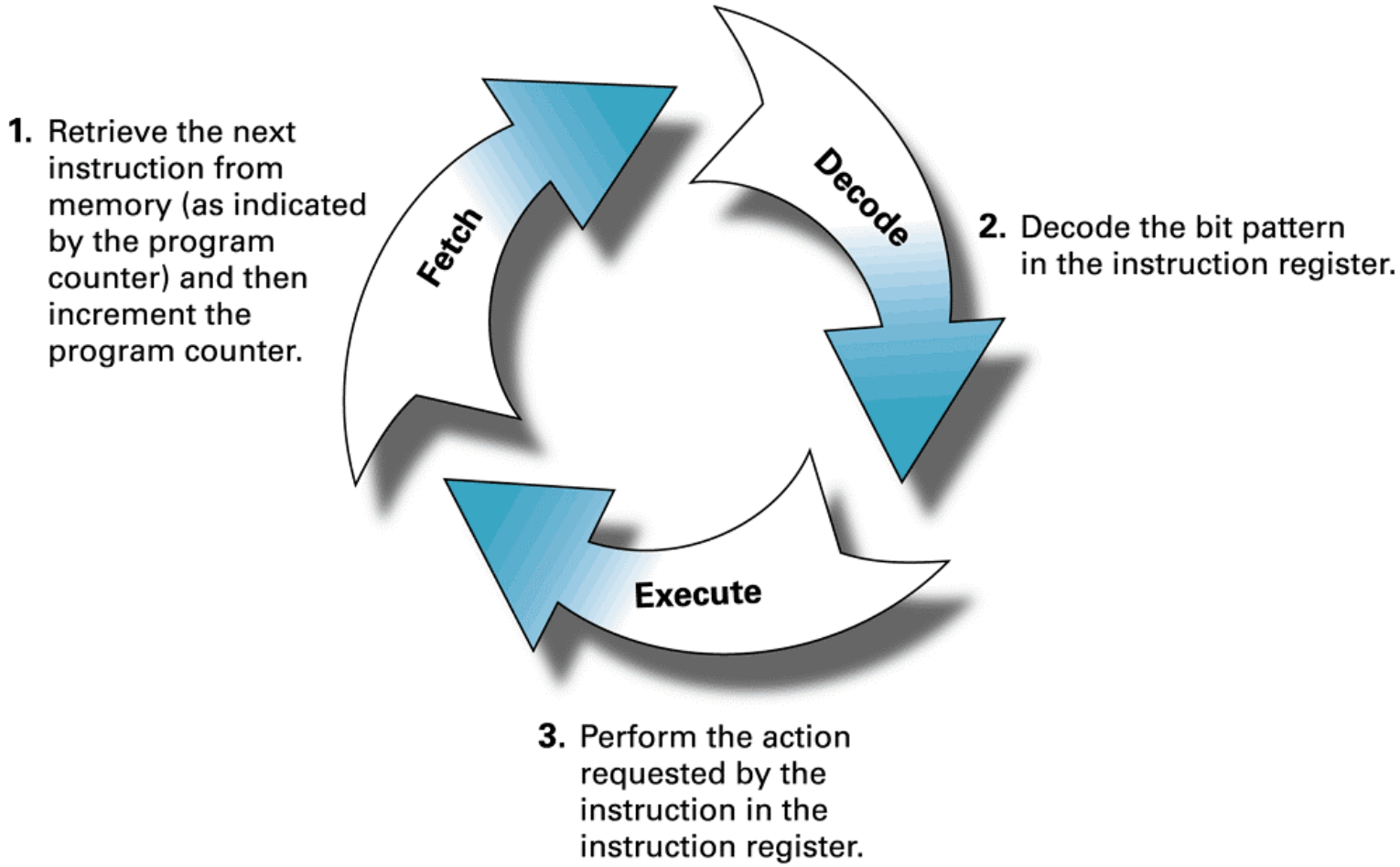
Example in Machine Language

Address	Contents	Address	Contents	Address	Contents
0D	B6	14	20	1B	0F
0E	5C	15	5A	1C	60
0F	5A	16	30	1D	12
10	20	17	0F	1E	30
11	5C	18	11	1F	0D
12	30	19	0E	20	C0
13	0E	1A	12	21	00

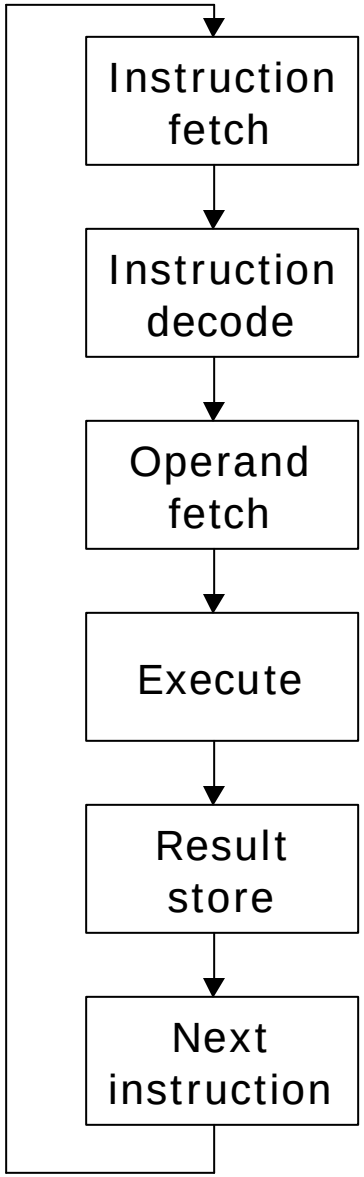
Register	
0	B6
1	5C
2	5A
3	

Op-code	Operand	Description
1	RXY	LOAD RegR from M(XY)
2	RXY	LOAD RegR with #XY
3	RXY	STORE M(XY) with RegR
5	RST	ADDI RegR := RegS + RegT
C	000	HALT execution
6	RST	ADDF RegR := RegS + RegT

The machine cycle



Execution Cycle



Obtain instruction from program storage

Determine required actions and instr. size

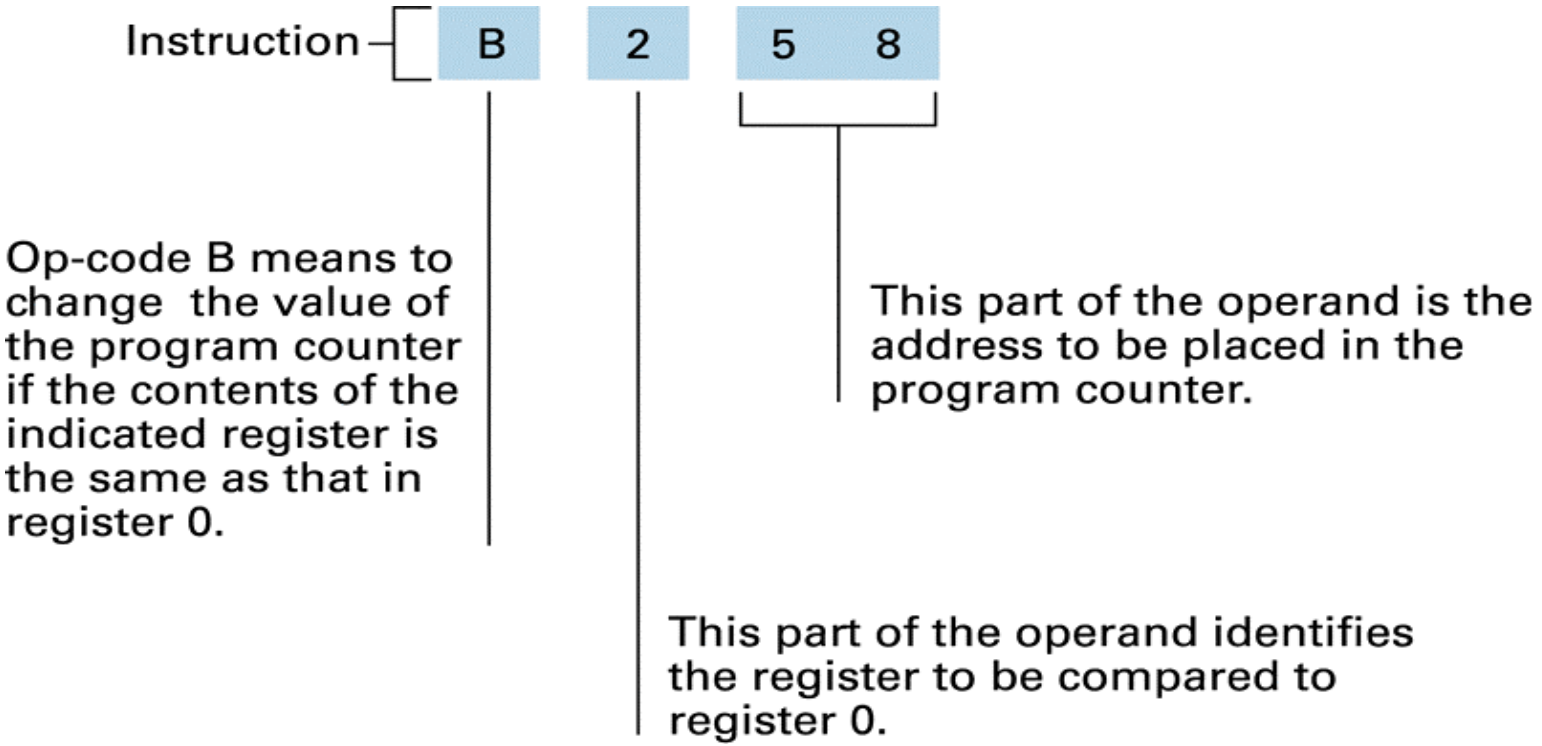
Locate and obtain operand data

Compute results in storage for later

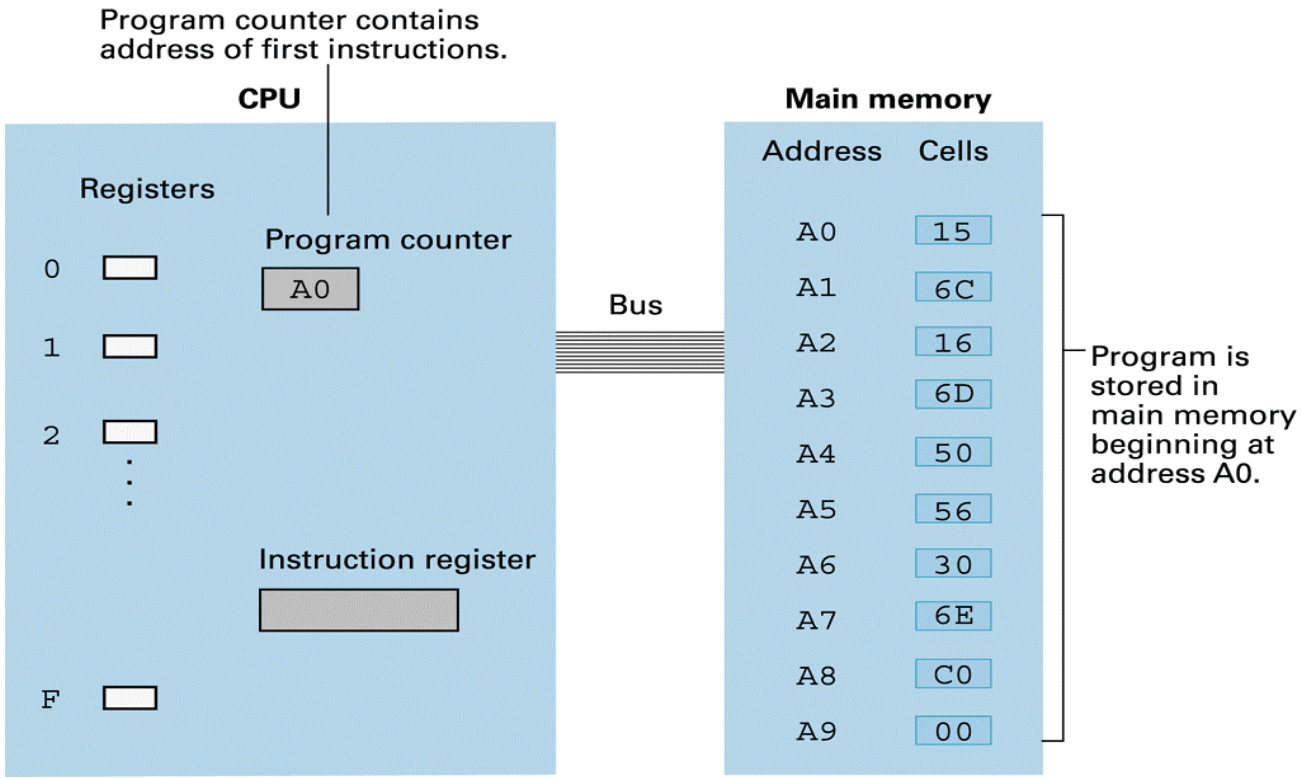
Deposit results in storage for later use

Determine successor instruction

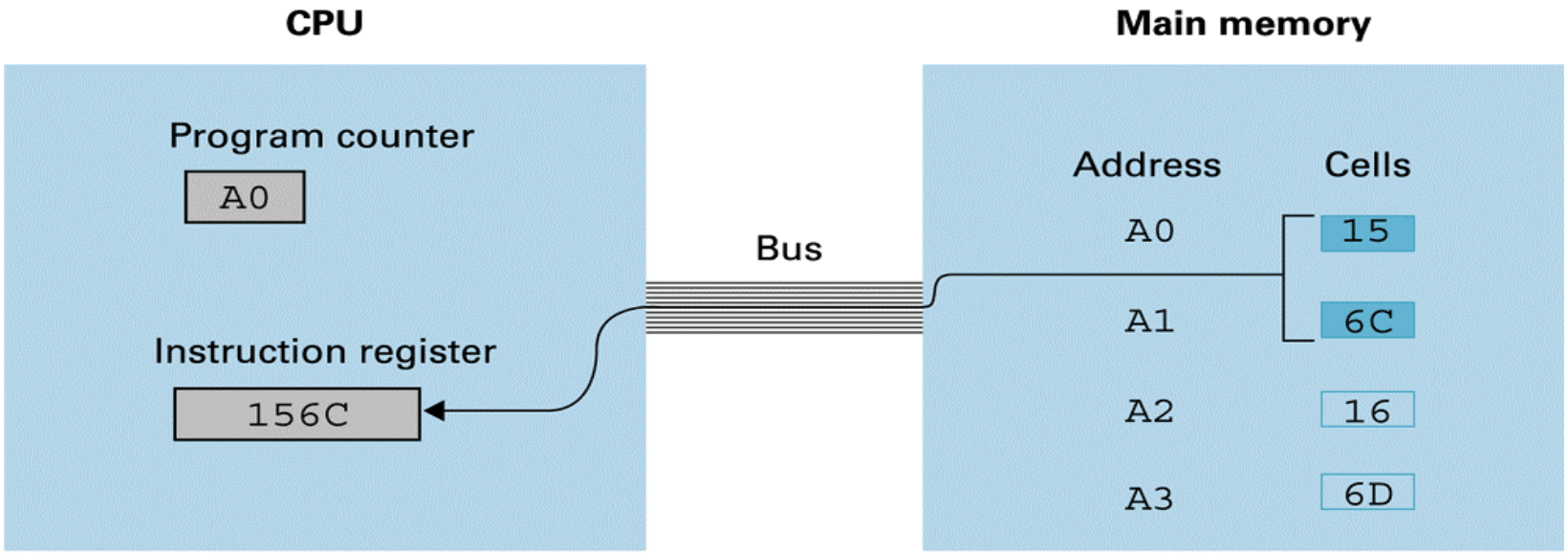
Decoding the instruction B258



Example 2 stored in main memory ready for execution

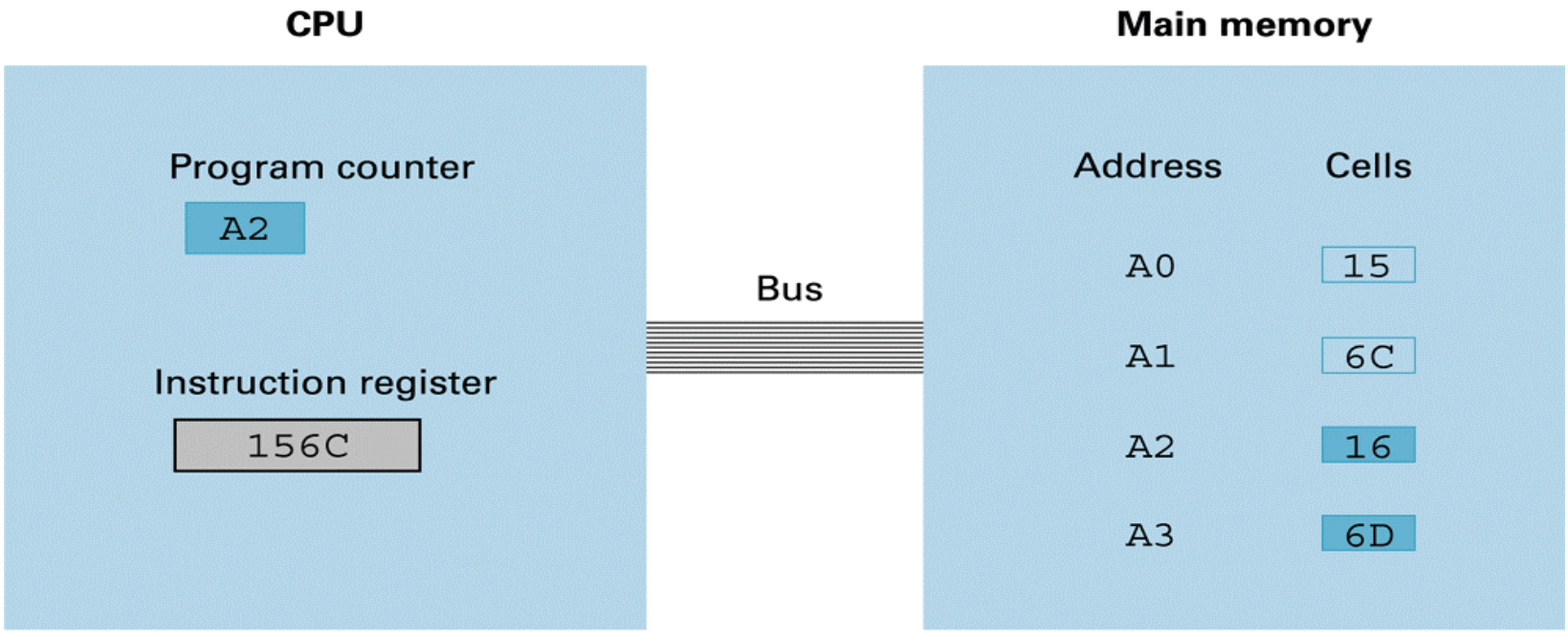


Performing the fetch step of the machine cycle (continued)



a. At the beginning of the fetch step the instruction starting at address A0 is retrieved from memory and placed in the instruction register.

Performing the fetch step of the machine cycle



b. Then the program counter is incremented so that it points to the next instruction.

. . .

- Pset 2 due tomorrow @ recitation
- Pset 3 on web page today
- Office hours this week
- Lecture Friday: Operating Systems
- Lectures next week: Tony Brogner (Draper)
- ?