Data Manipulation

Işıl ÖZ, IZTECH, Fall 2023

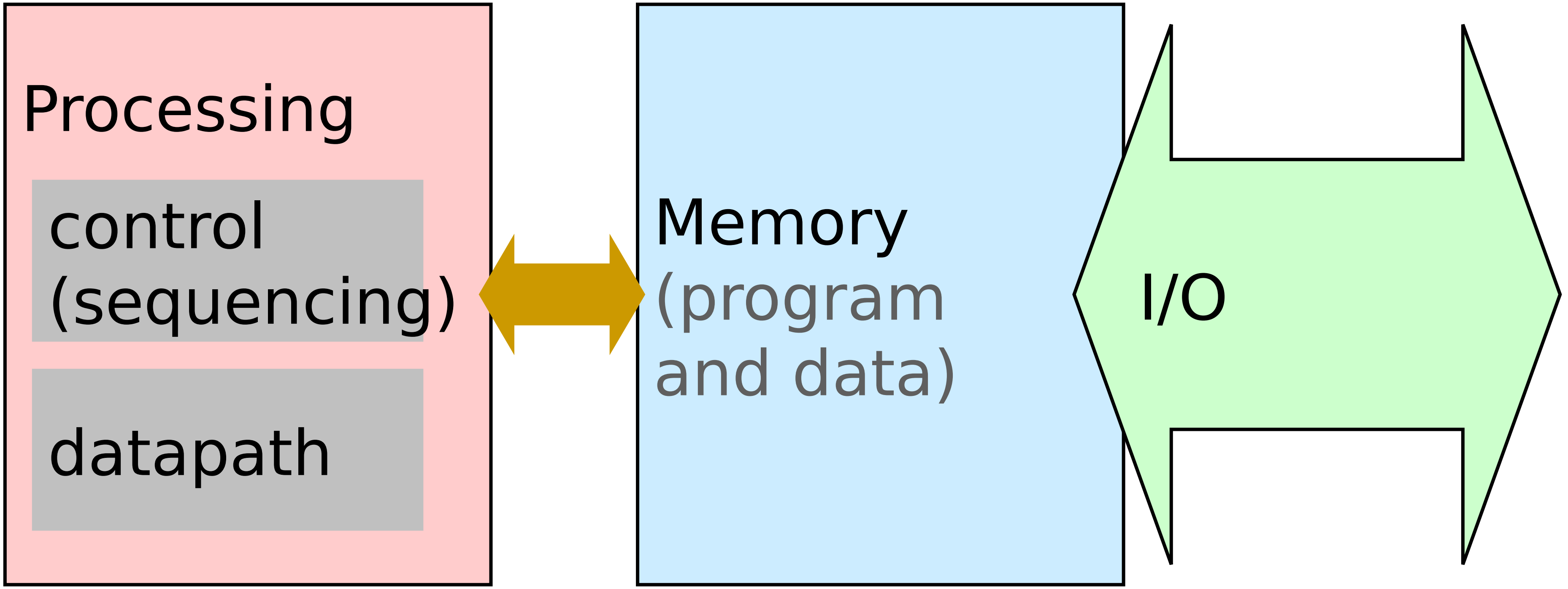
18 October 2023

Data Manipulation

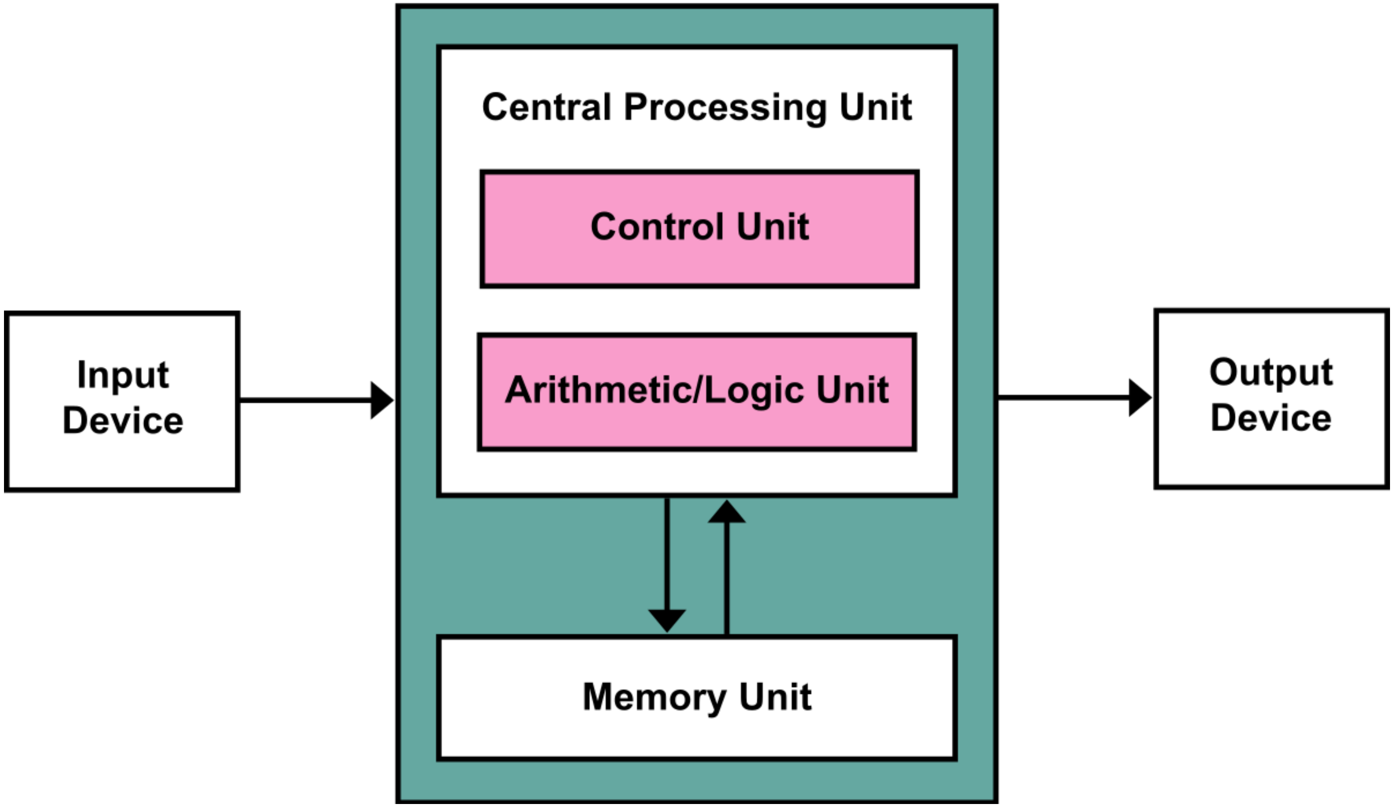
Moving data from one location to another

Performing operations such as arithmetic calculations, text editing, and image manipulation

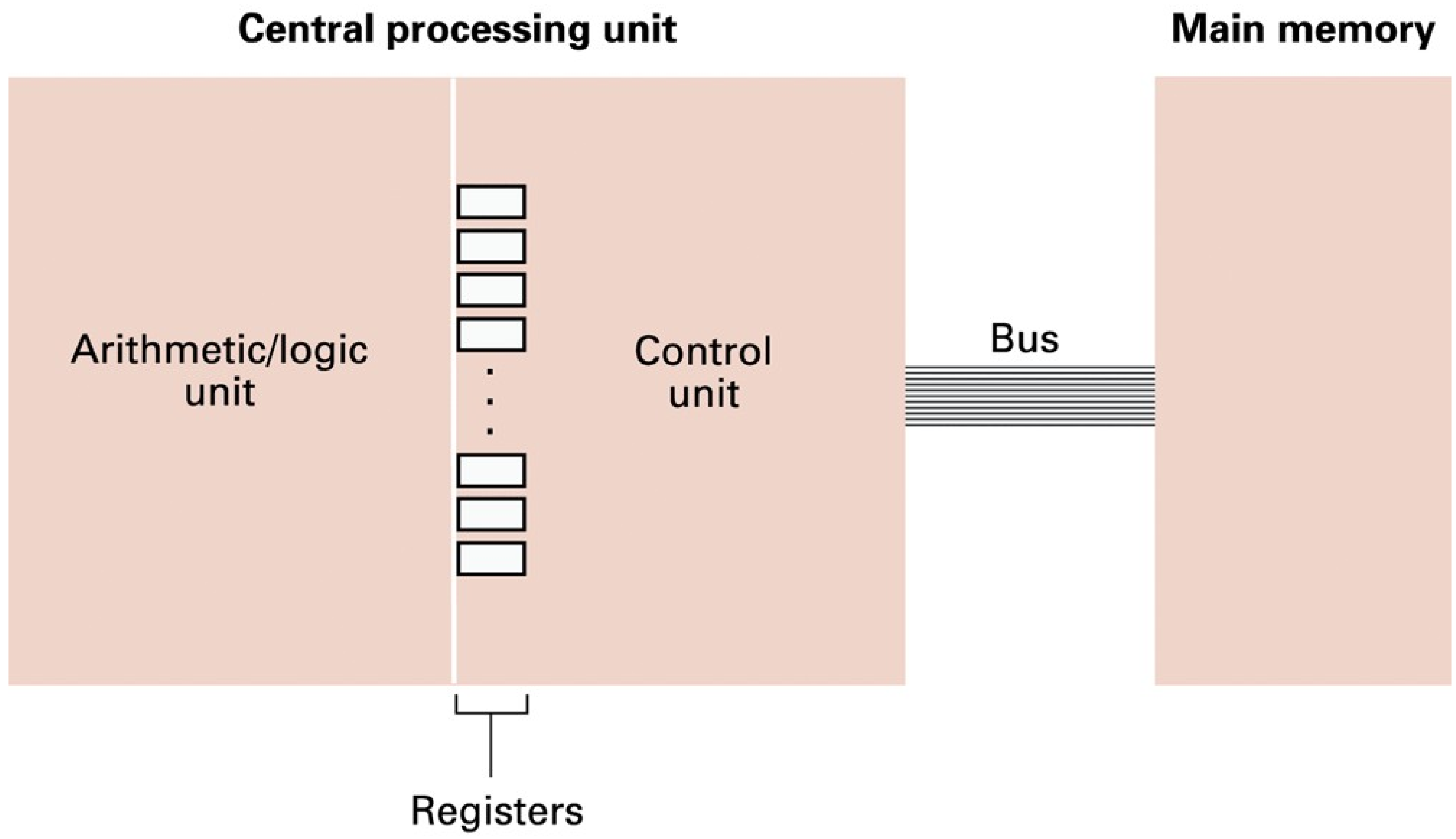
Computer System



The von Neumann Model



CPU and Main Memory



Central Processing Unit (CPU)

Processor to perform computations

Executes a sequence of stored instructions called a program, program is kept in the main memory

Control unit : responsible for deciding which instruction in a program should be executed

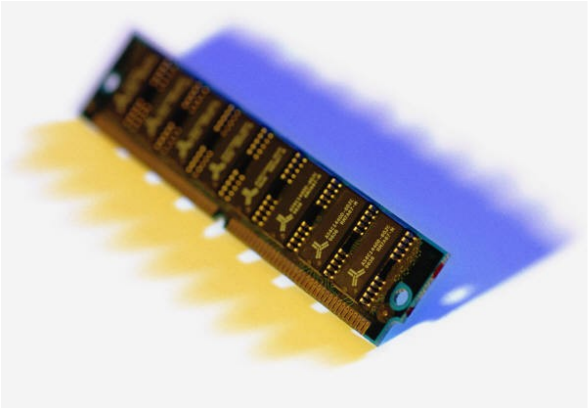
Arithmetic and logic unit (ALU) : responsible for executing the actual instructions

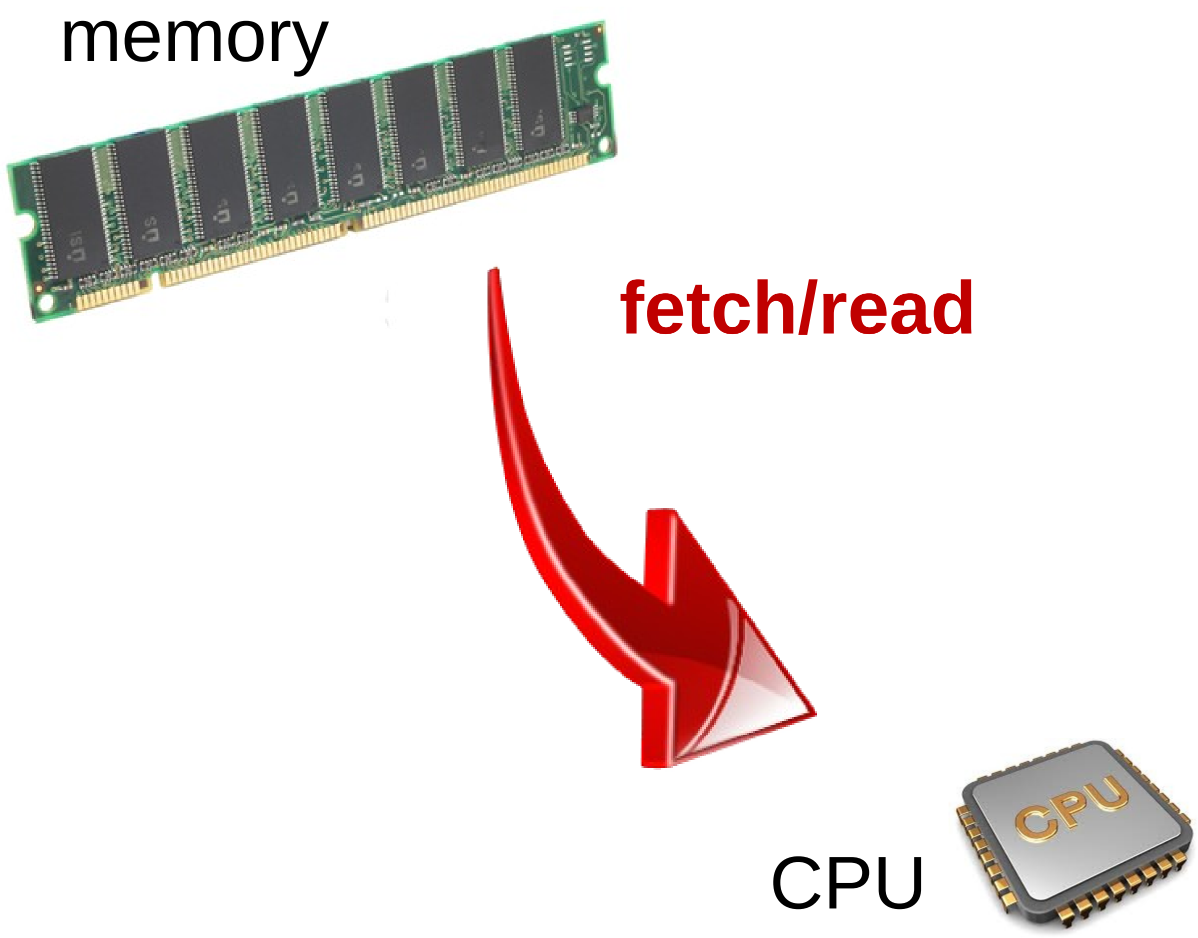
Register : quickly accessible location available to CPU

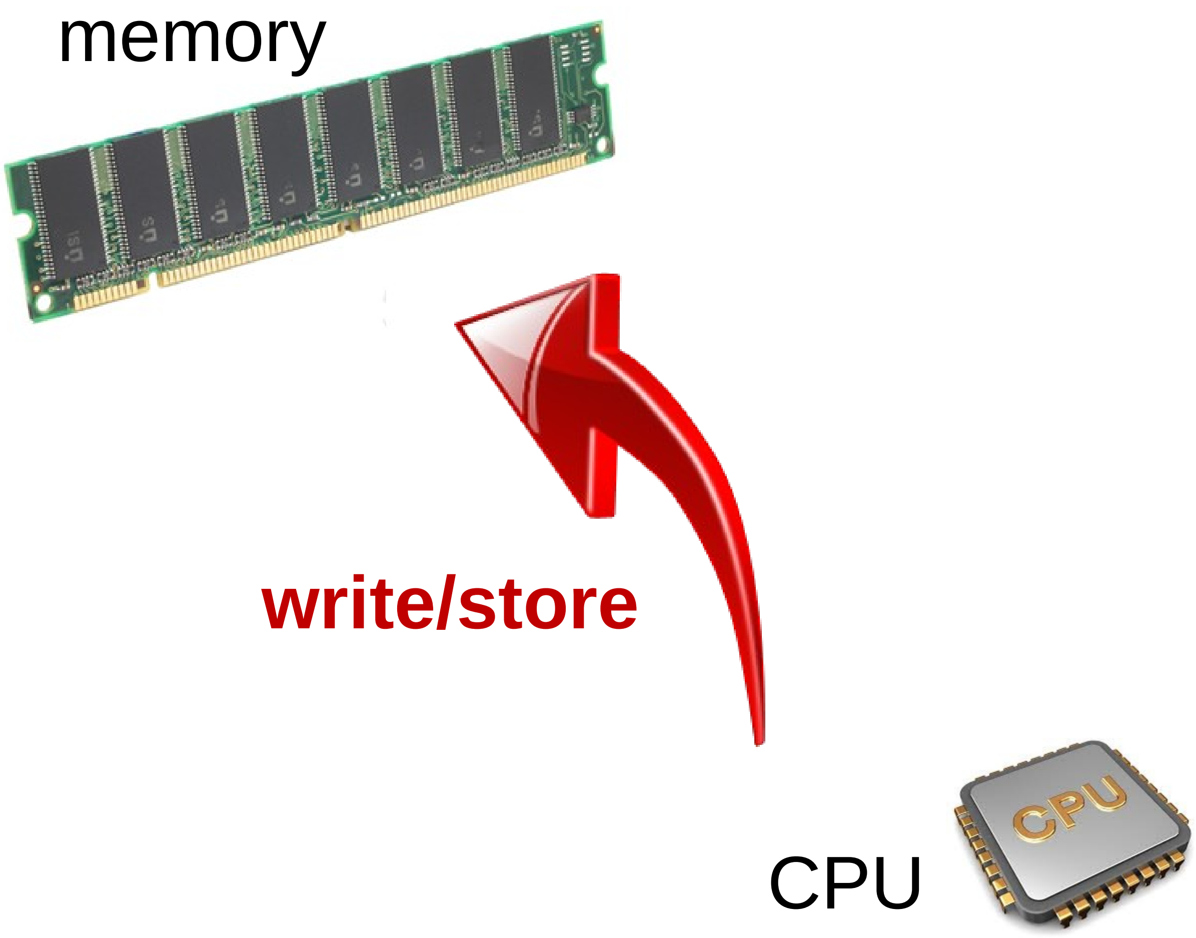
Main Memory

Collection of locations, each of which is capable of storing both instructions and data

Every location consists of an address, which is used to access the location, and the contents of the location




memory
fetch/read
CPU
CPU


memory
write/store
CPU
CPU

Example Program

Compute sin(x) using taylor expansion: sin(x) = x – x^3/3! + x^5/5! + x^7/7! + ...

```
void sinx(int N, int terms, float* x, float* result)
{
   for (int i=0; i<N; i++)
   {
      float value = x[i];
      float numer = x[i] * x[i] * x[i];
      int denom = 6;  // 3!
      int sign = -1;

      for (int j=1; j<=terms; j++)
      {
         value += sign * numer / denom
         numer *= x[i] * x[i];
         denom *= (j+3) * (j+4);
         sign *= -1;
      }

      result[i] = value;
   }
}
```

Compile Program

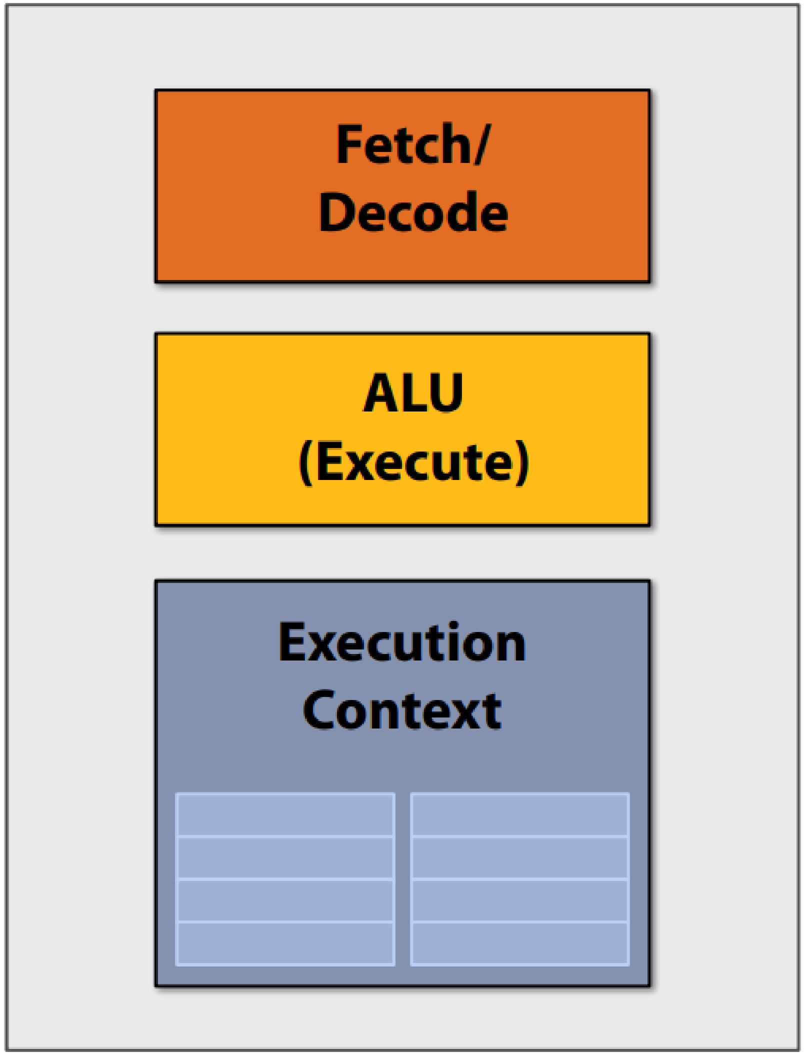
```c
void sinx(int N, int terms, float* x, float* result)
{
   for (int i=0; i<N; i++)
   {
      float value = x[i];
      float numer = x[i] * x[i] * x[i];
      int denom = 6;  // 3!
      int sign = -1;

      for (int j=1; j<=terms; j++)
      {
         value += sign * numer / denom
         numer *= x[i] * x[i];
         denom *= (j+3) * (j+4);
         sign *= -1;
      }

      result[i] = value;
   }
}
```

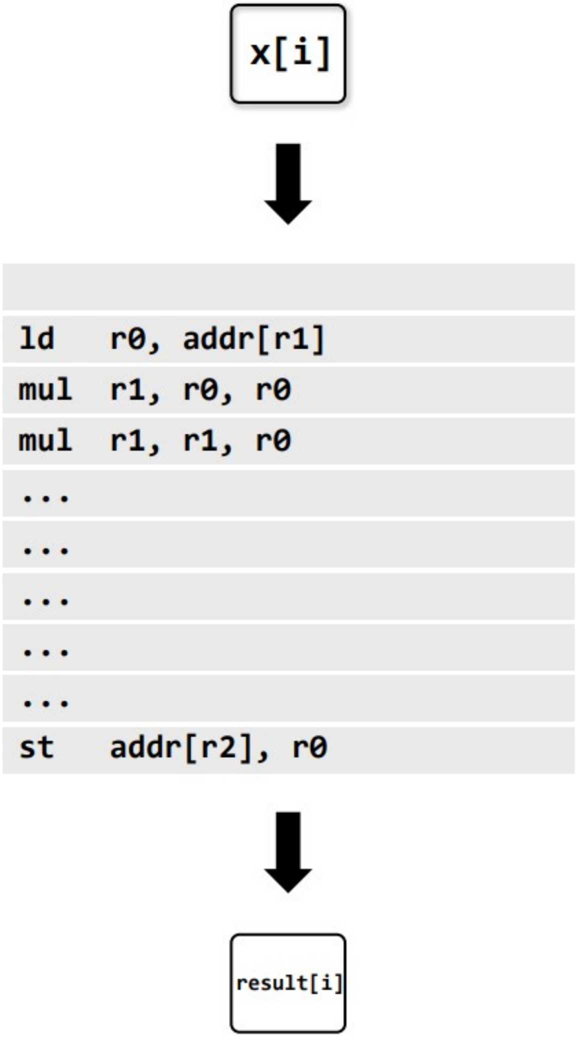
x[i]

```
ld    r0, addr[r1]
mul   r1, r0, r0
mul   r1, r1, r0
...
...
...
...
...
st    addr[r2], r0
```

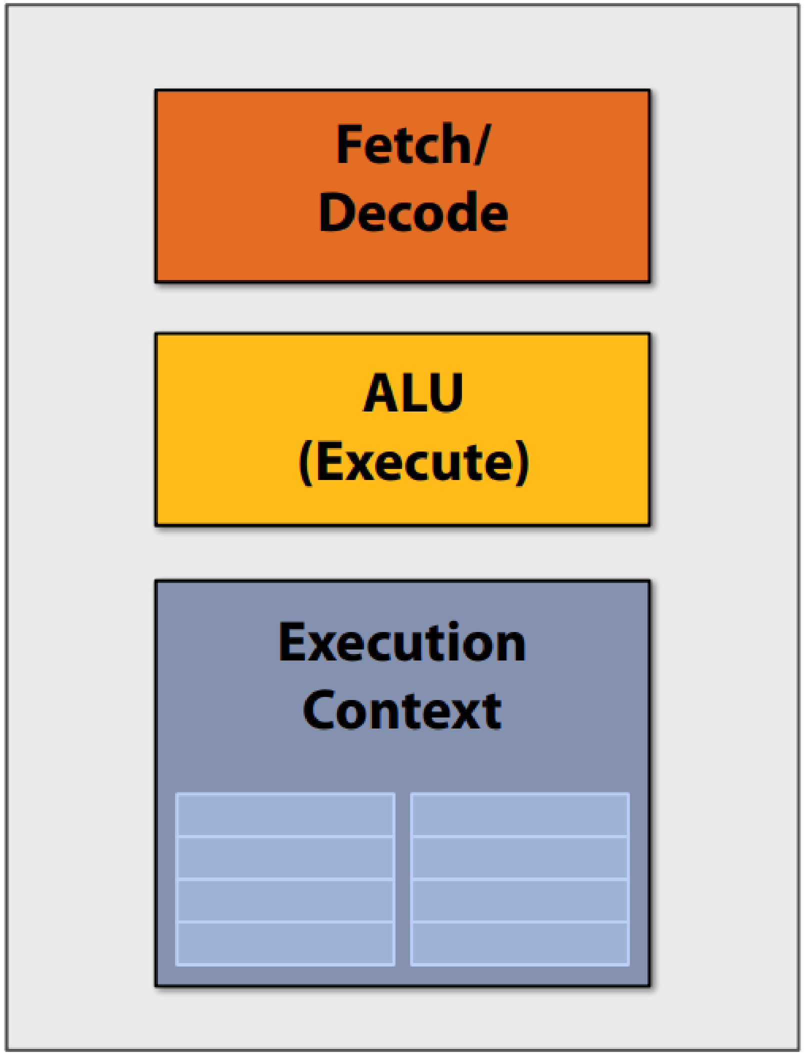
result[i]

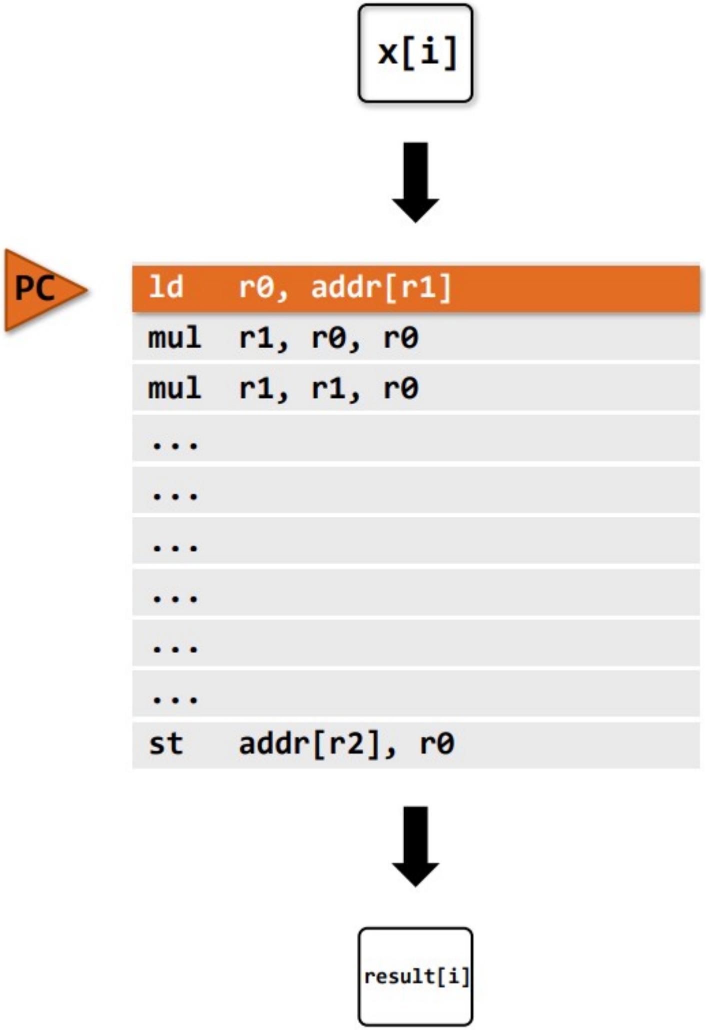
Execute Program



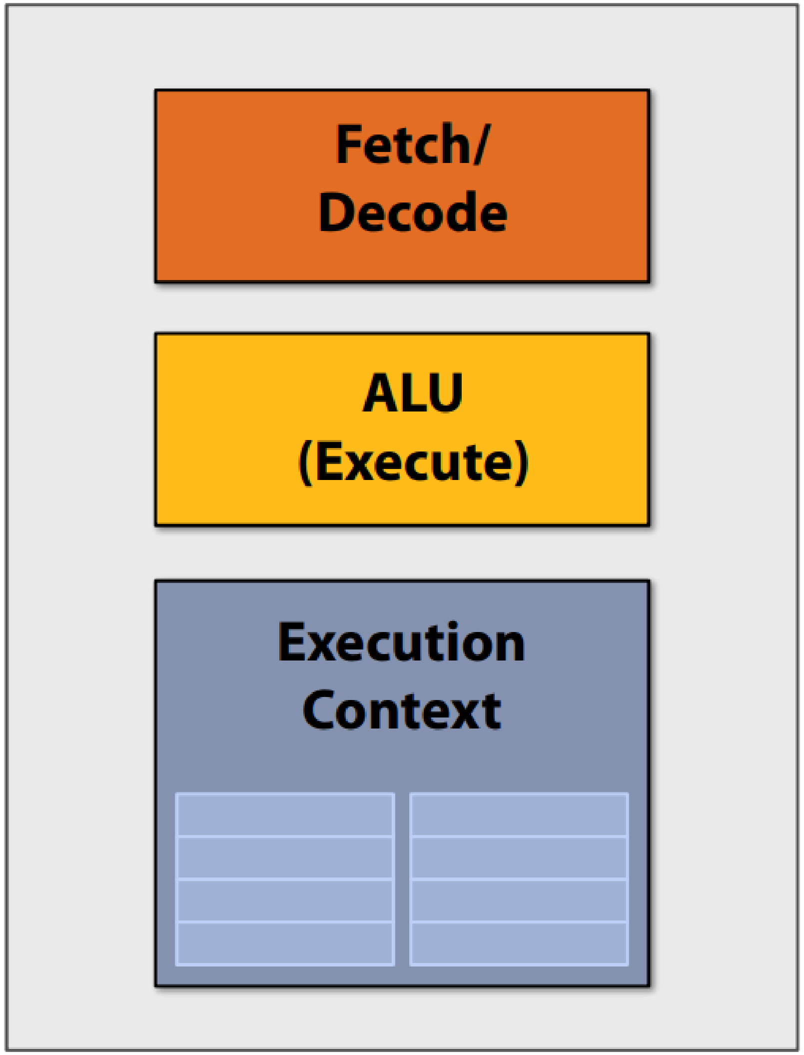


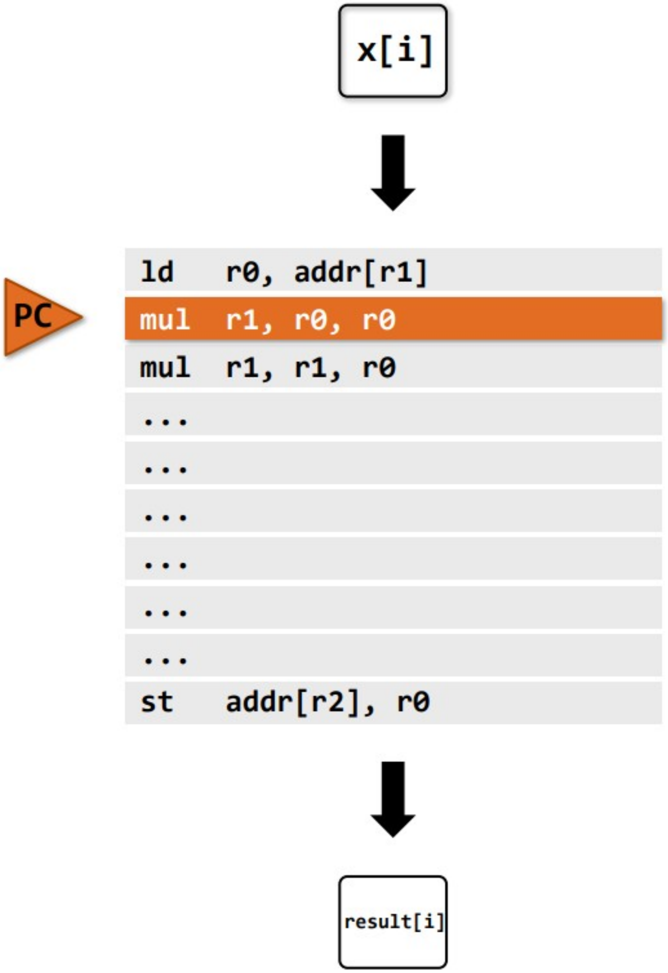
Execute One Instruction per clock



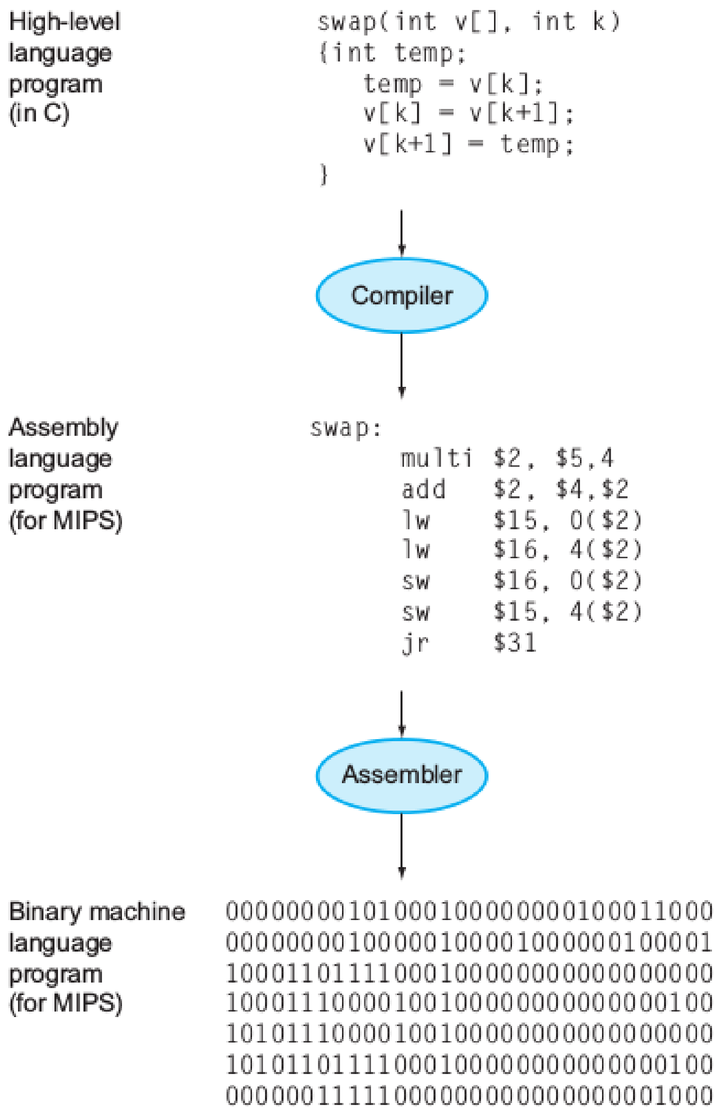


Execute One Instruction per clock

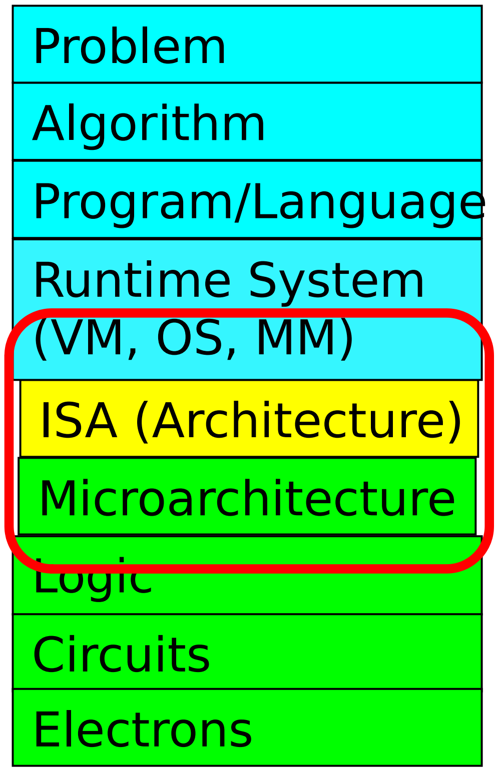




From a High-Level Language to the Language of Hardware



Levels of Transformation



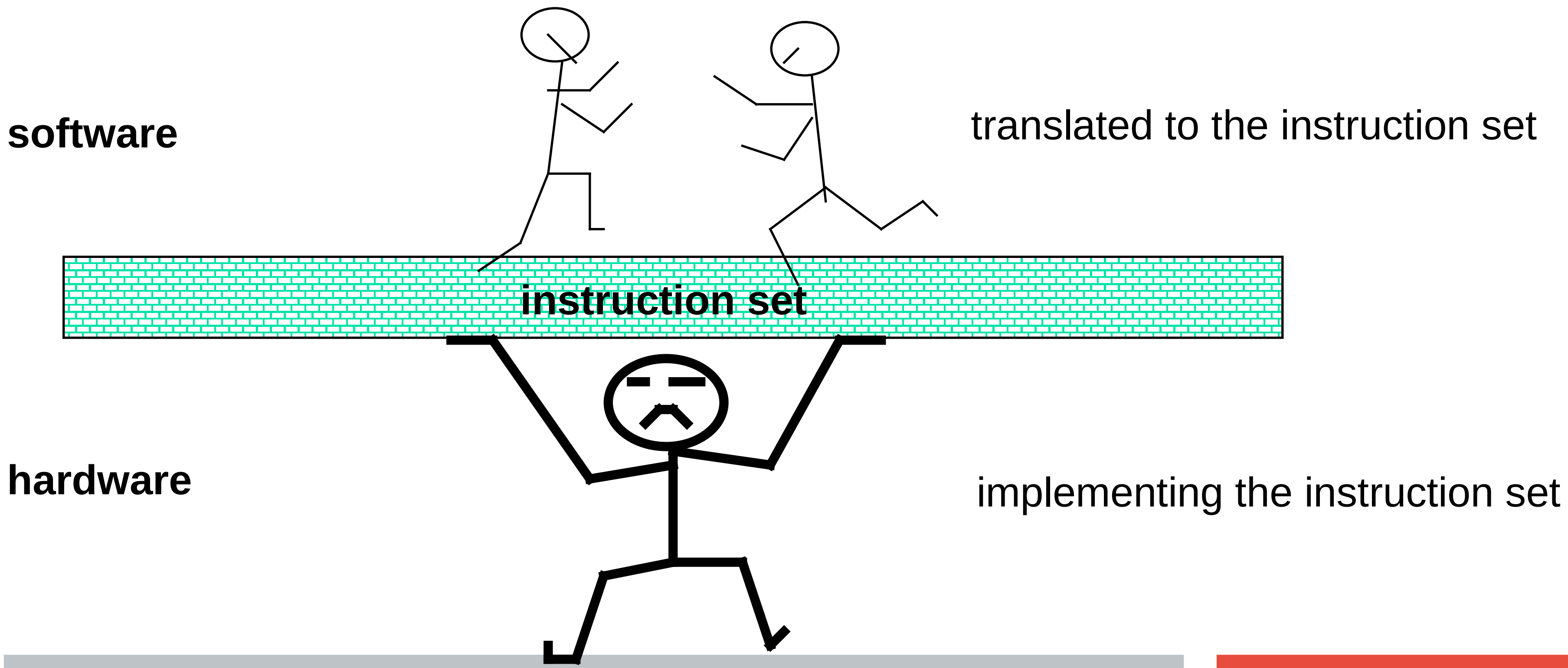
Patt, "Requirements, Bottlenecks, and Good Fortune: Agents for Microprocessor Evolution," Proceedings of the IEEE 2001.

Instruction Set

Computer's language: instructions (in assembly)

Its vocabulary: instruction set (e.g., x86, MIPS)

Interface between processor and low-level software



Machine Language

The set of all instructions recognized by a machine

Machine instruction: An instruction (or command) encoded as a bit pattern recognizable by the CPU

Machine Instruction Types

Data Transfer: copy data from one location to another

LOAD-READ from memory, STORE-WRITE into memory

Arithmetic/Logic: perform arithmetic/logic operations on the values

ADD, SUBTRACT, MULTIPLY, DIVIDE, AND, OR

Control: direct the execution of the program

JUMP, BRANCH

Simple Machine Language (Complete Given in Appendix C)

Op-code	Operand	Description
1	RXY	LOAD the register R with the bit pattern found in the memory cell whose address is XY. *Example:* 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	LOAD the register R with the bit pattern XY. *Example:* 20A3 would cause the value A3 to be placed in register 0.
3	RXY	STORE the bit pattern found in register R in the memory cell whose address is XY. *Example:* 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	MOVE the bit pattern found in register R to register S. *Example:* 40A4 would cause the contents of register A to be copied into register 4.
5	RST	ADD the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. *Example:* 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.
6	RST	ADD the bit patterns in registers S and T as though they represented values in floating-point notation and leave the floating-point result in register R. *Example:* 634E would cause the values in registers 4 and E to be added as floating-point values and the result to be placed in register 3.
7	RST	OR the bit patterns in registers S and T and place the result in register R. *Example:* 7CB4 would cause the result of ORing the contents of registers B and 4 to be placed in register C.

Language Features

16-bit instructions

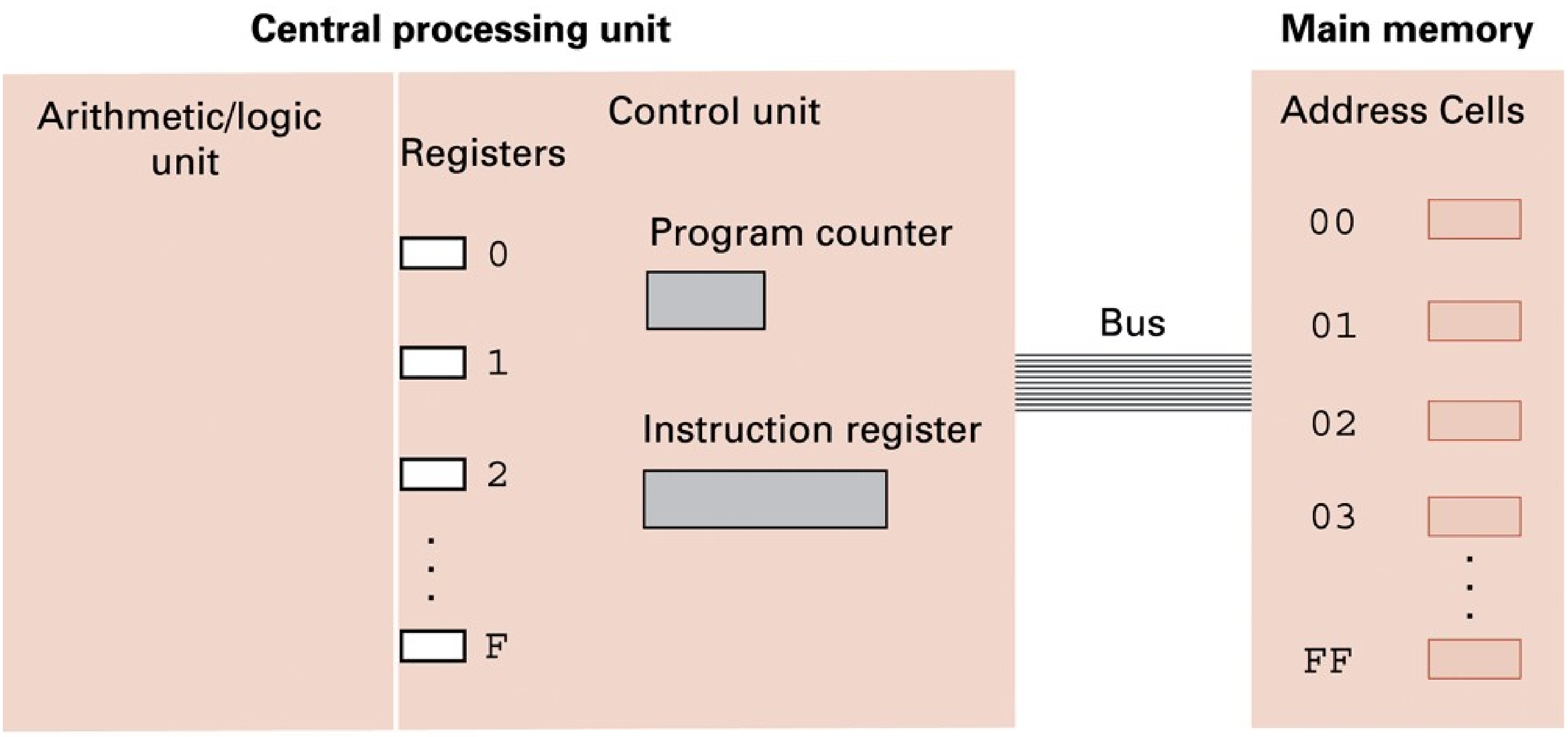
16 general-purpose registers (4-bits register number, 8-bits value)

256 cells in main memory (addressed by 8-bits)

Special-purpose registers

- Program counter: address of next instruction
- Instruction register: current instruction

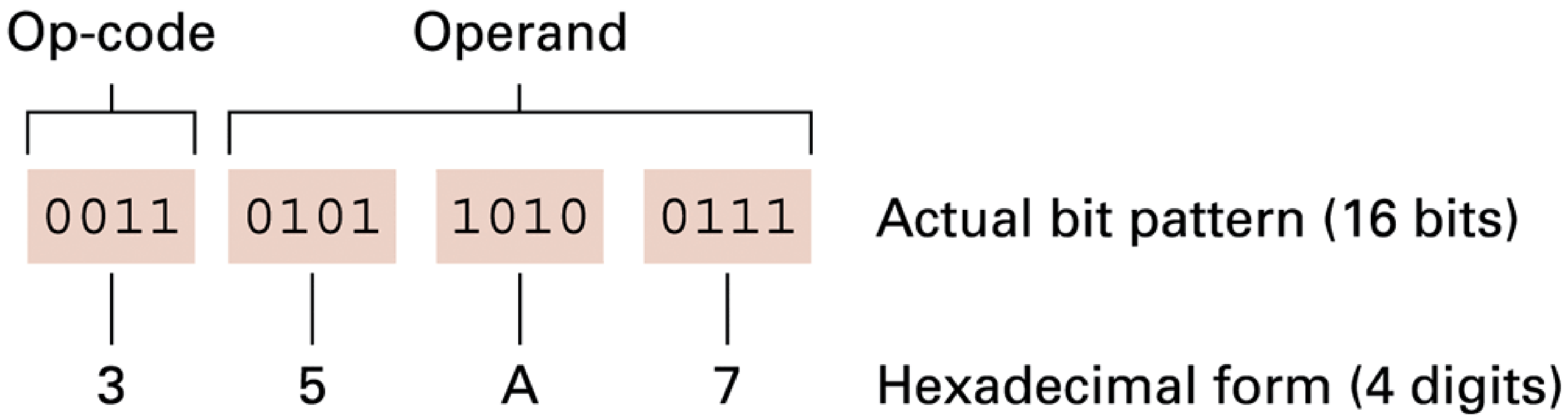
Simple Machine Language (Given in Appendix C)



The Composition of an Instruction

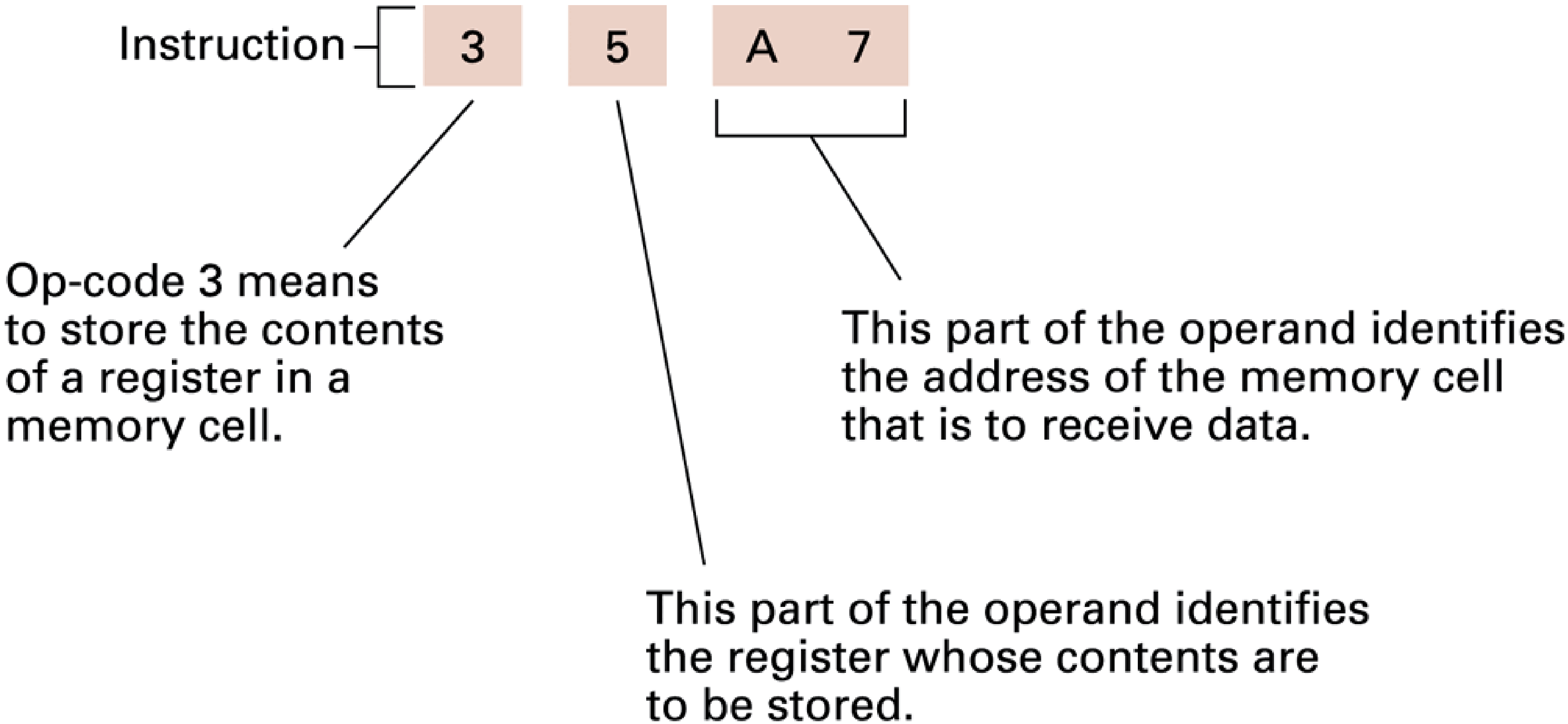
Op-code: Specifies which operation to execute

Operand: Gives more detailed information about the operation

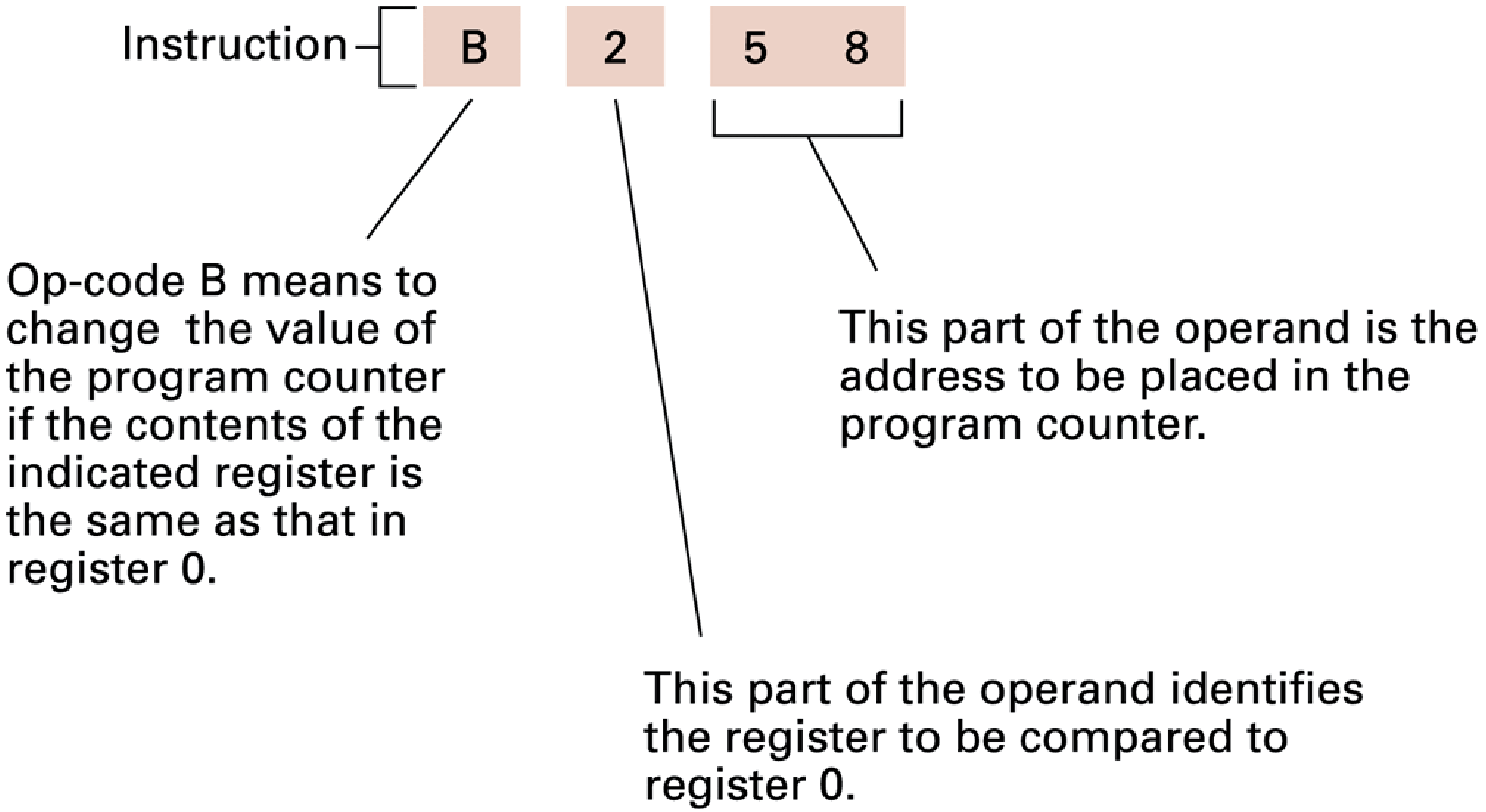


Example: Decoding an Instruction

STORE the value in register 5 in the memory cell whose address is A7



Example: Decoding an Instruction



Example: Encoding Instructions

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.

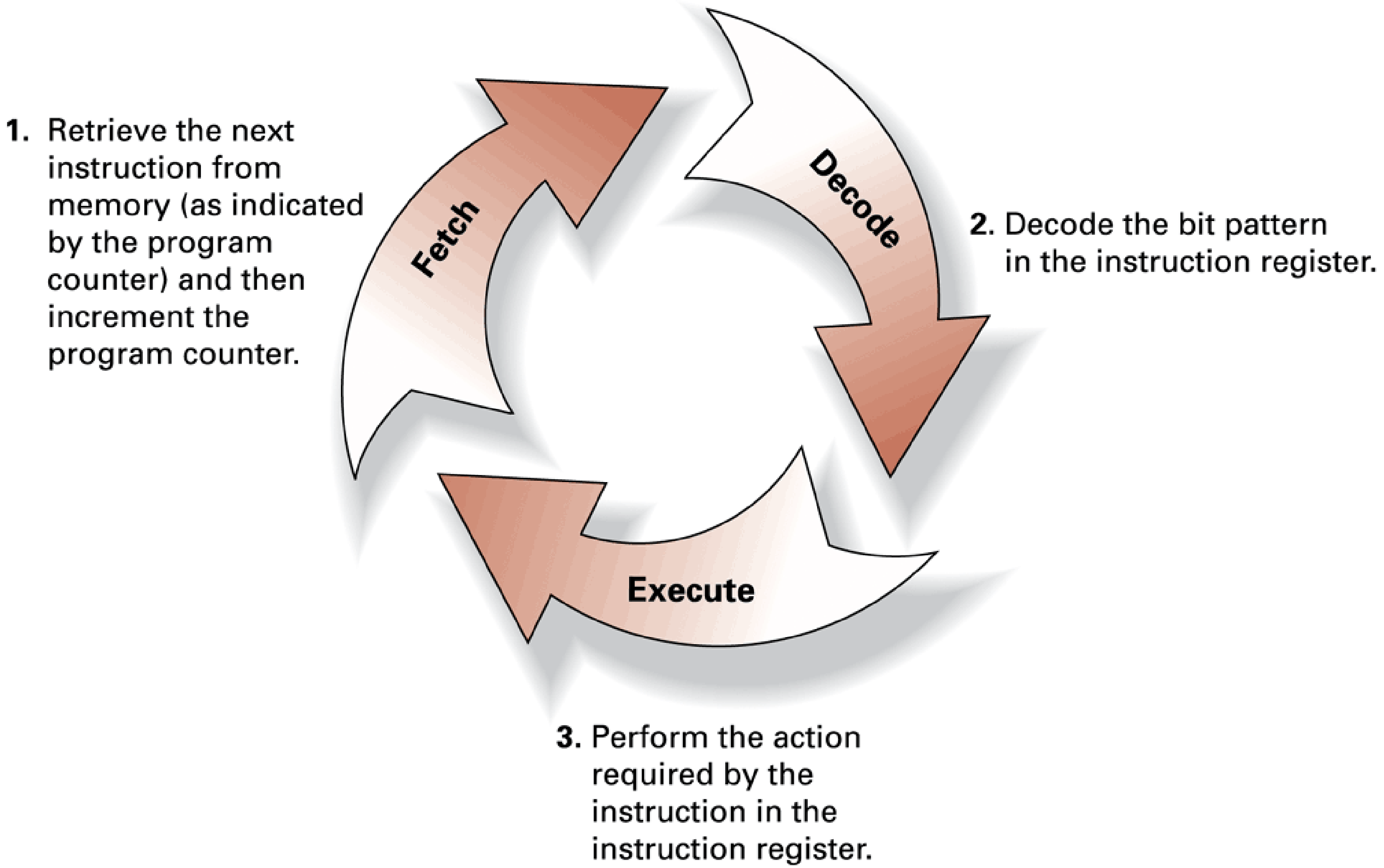
Step 3. Activate the addition circuitry with the registers used in Steps 1 and 2 as inputs and another register designated to hold the result.

Step 4. Store the result in memory.

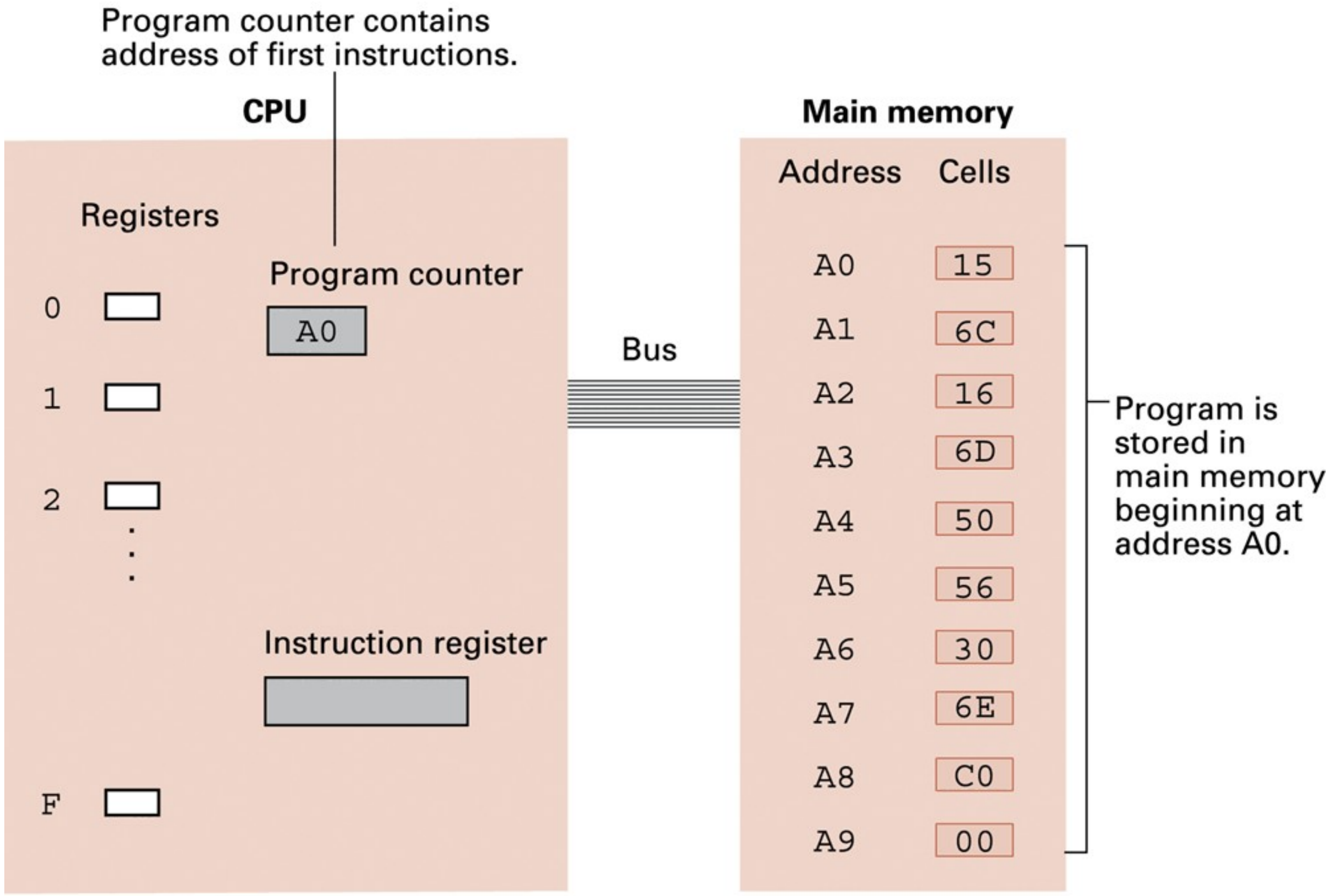
Step 5. Stop.

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.

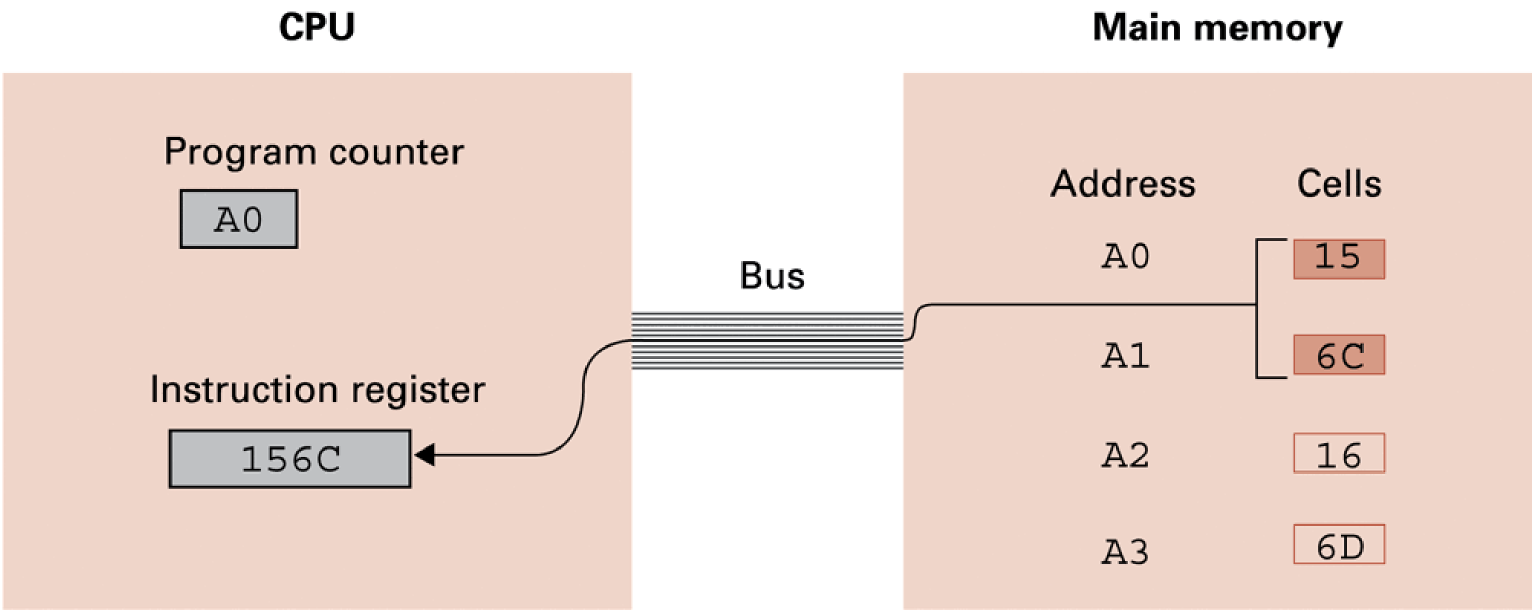
Program Execution



Example: Program Execution

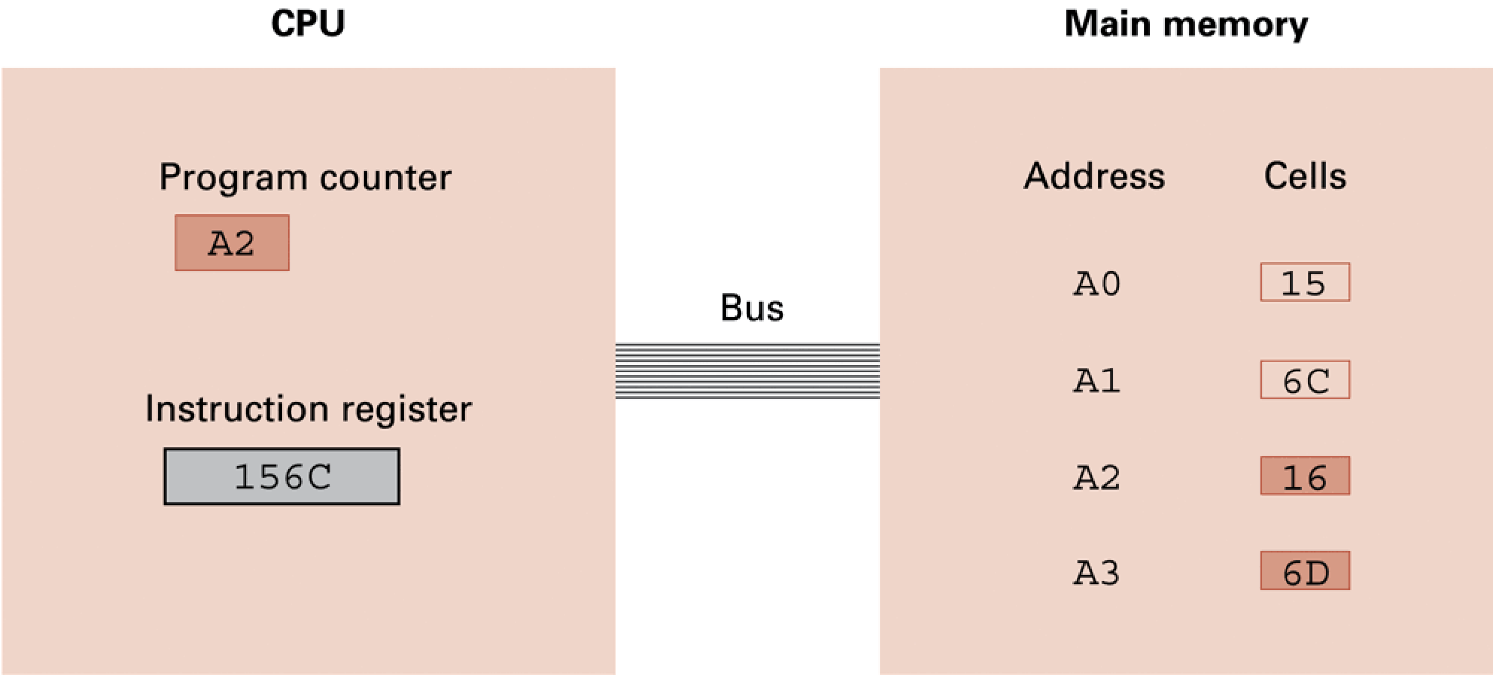


Fetch Stage



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

Fetch Stage



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

Decode Stage/Execute Stage

Instruction 1 5 6C

Load register 5 with the value in memory cell addressed by 6C (read the memory address 6C into register 5)

Decode: Understand this instruction

Execute: Load the value in memory into register

Example Execution

Address	Contents
00	20
01	04
02	21
03	01
04	40
05	12
06	51
07	12
08	B1
09	0C
0A	B0
0B	06
0C	C0
0D	00

Assume that the machine starts with its program counter containing 00.

Recommended Lectures

Spring 2023, Digital Design and Computer Architecture - Lecture 1: Introduction and Basics, ETH Zurich, by Onur Mutlu
https://www.youtube.com/watch?v=VcKjvwD930o

2017 ACM A.M. Turing Award Lecture, John Hennessy and David Patterson

https://www.youtube.com/watch?v=3LVeEjsn8Ts