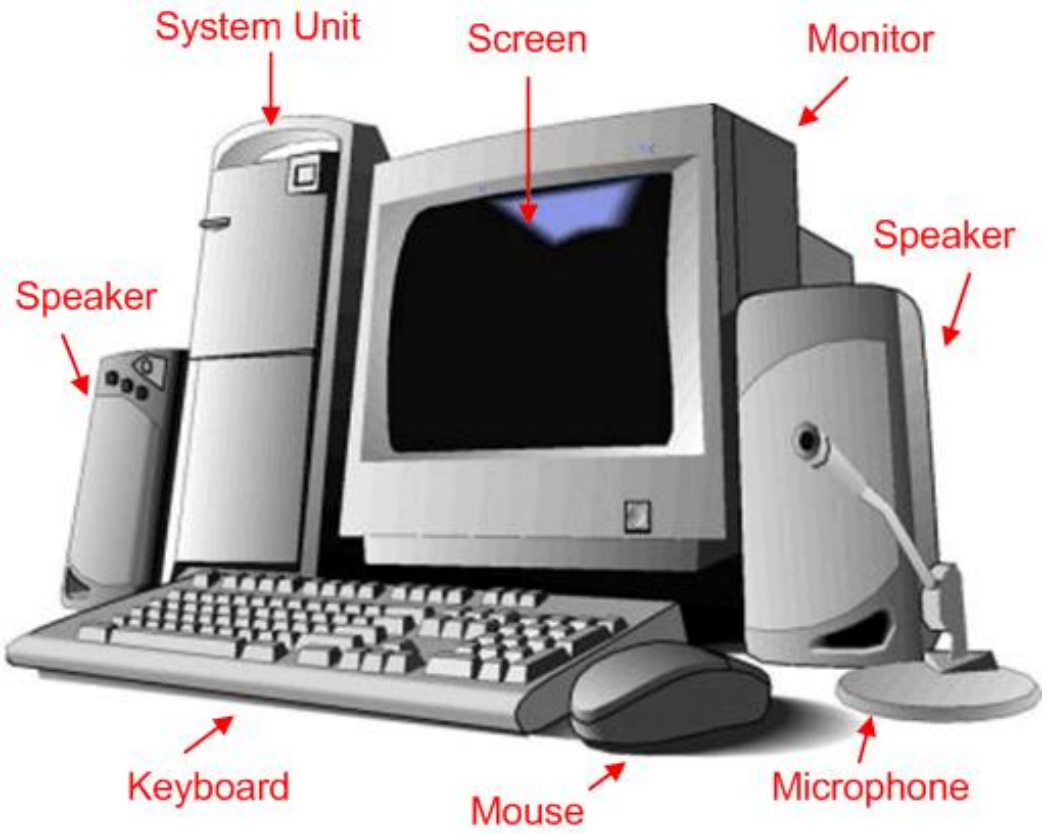
Chapter 1 Introduction to computer

* background and history
* review number systems
* review basic logic circuits
* structure of computer
* computer operation

1.1 Background

What is a Computer?

A **computer** is a programmable machine that receives input, stores and manipulates data//information, and provides output in a useful format.

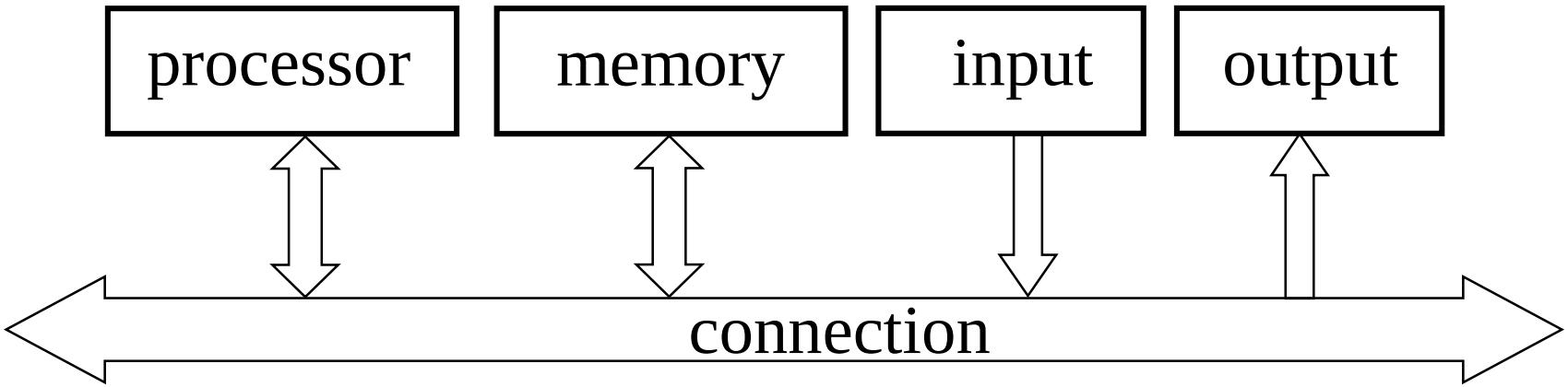


Computer has:

1. Processor to execute programs
2. Memory to store programs and data
3. Mechanism for transferring data to and from the outside world
4. Software, e.g. Operating System, application programs

Hardware + Software

Top-level view of computer



Processor = Central Processing Unit (CPU):
registers
Arithmetic Logic Unit (ALU)
control unit

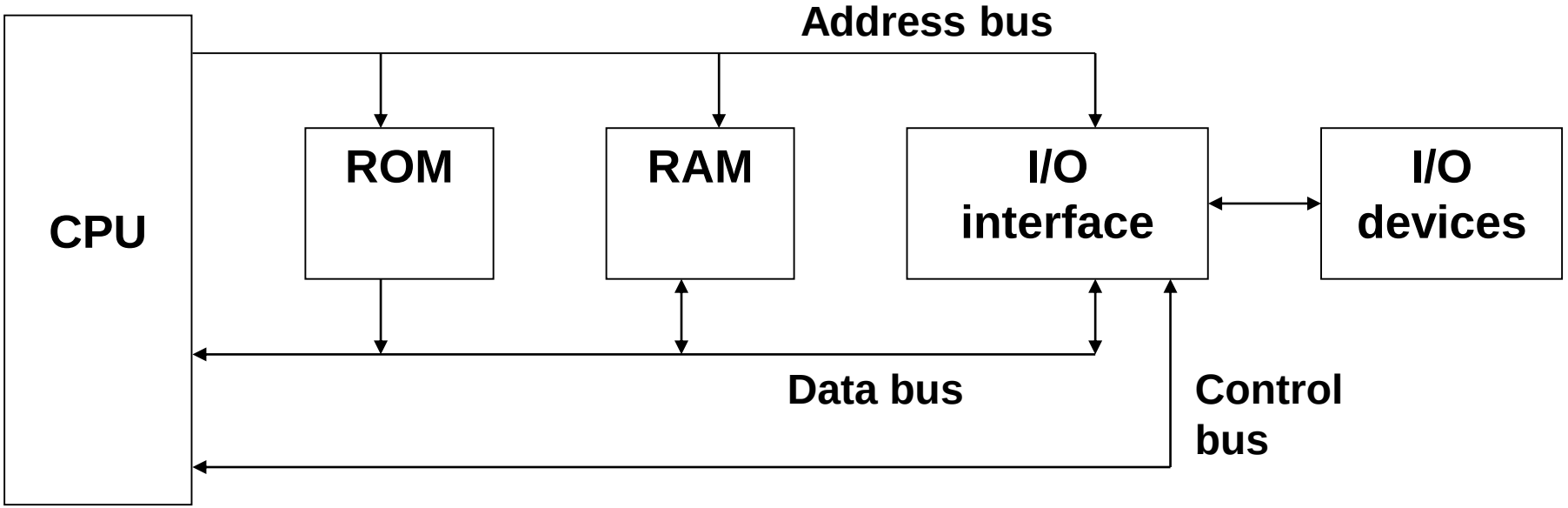
BLOCK DIAGRAM OF A BASIC COMPUTER SYSTEM

Basic computer system consists of a Central Processing Unit (CPU), memory (RAM and ROM), Input/Output (I/O) units.

CPU: execute the programs

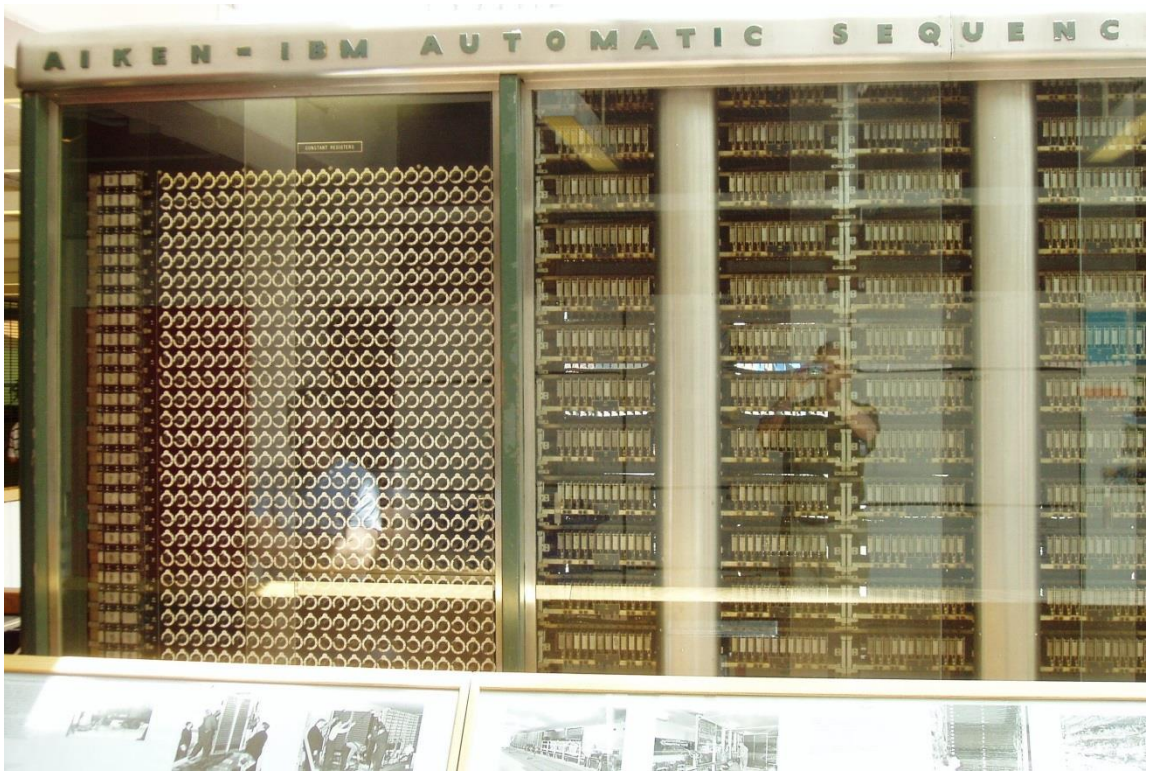
Memory: store data and programs

I/O: communicate with outside world



Block diagram of a basic computer system

In old days, computers were very large!



What is a microcomputer?

A **micro**computer is a small, relatively inexpensive computer with a **micro**processor as its central processing unit (CPU).

Desktop personal computer (PC)

Laptop computer

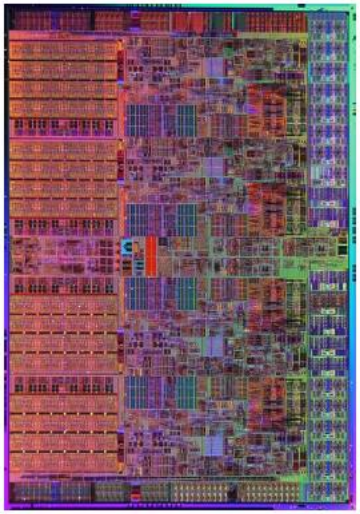
Notebook computer

Tablet

What is a microprocessor?

Microprocessor is a Central Processing Unit (CPU) on a single chip.

It contains millions of transistors connected by wires.



Core i7 die

Picture: Intel

Core i7 in package

Picture: Ebbesen

1.2 Evolution of computer

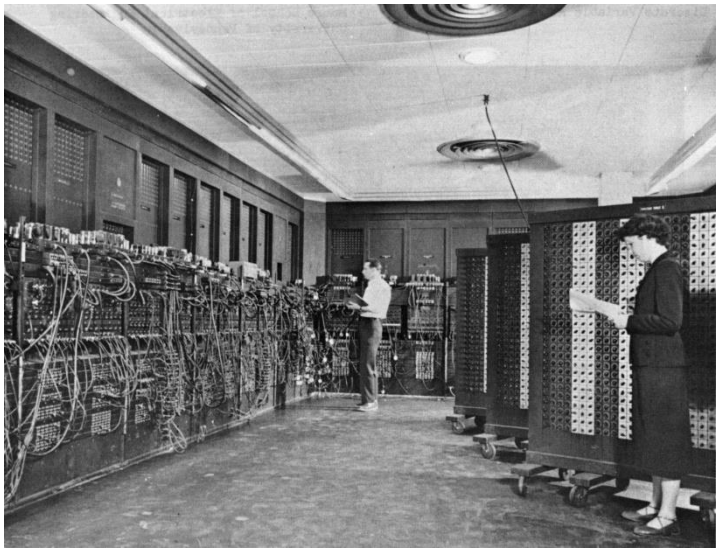
- computers have been developed over 70 years

- evolution of computers has been characterized by:

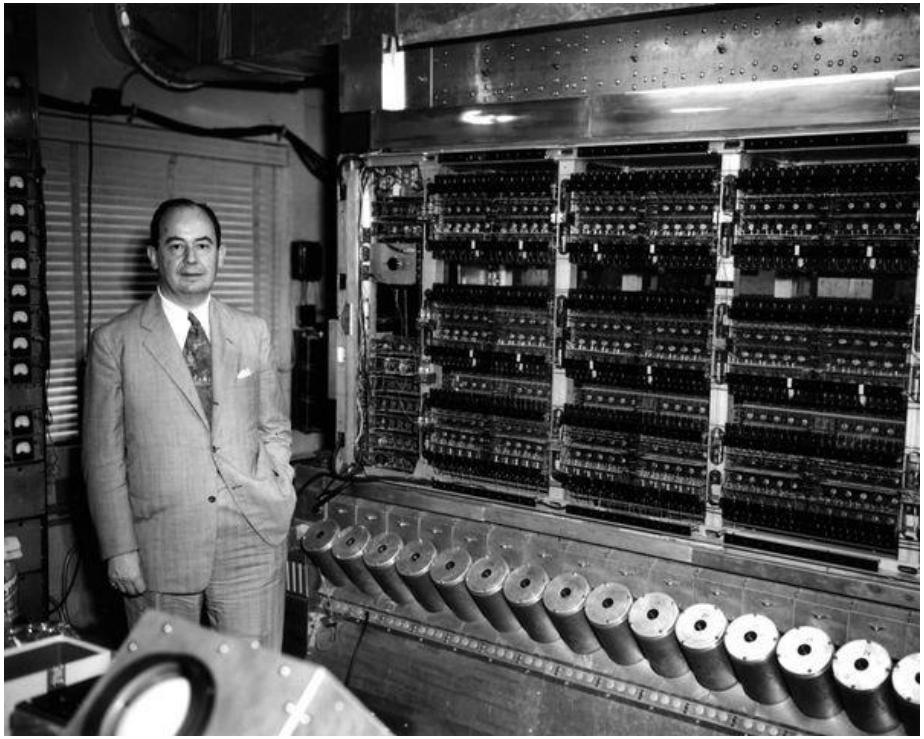
increasing processor speed
decreasing computer size
increasing memory size
increasing I/O capacity and speed

1.2.1 first generation – vacuum tube (1946-1957)

- ENIAC
  - numbers were represented in decimal form (10 vacuum tubes for one digit)
  - programmed manually by setting switches and plugging/unplugging cables

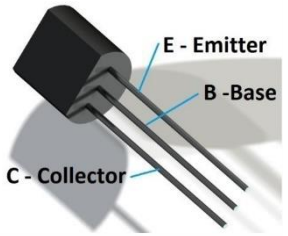


- von Neumann machine (IAS computer)
  - numbers were represented in binary form
  - program could be stored in memory (mercury delay-line)

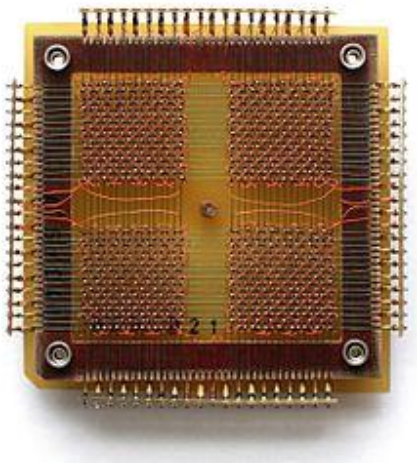


1.2.2 second generation – transistor (1957-1964)

- transistor is smaller, cheaper and dissipates less heat than vacuum tube



- magnetic core memory
- high-level programming languages
- system software



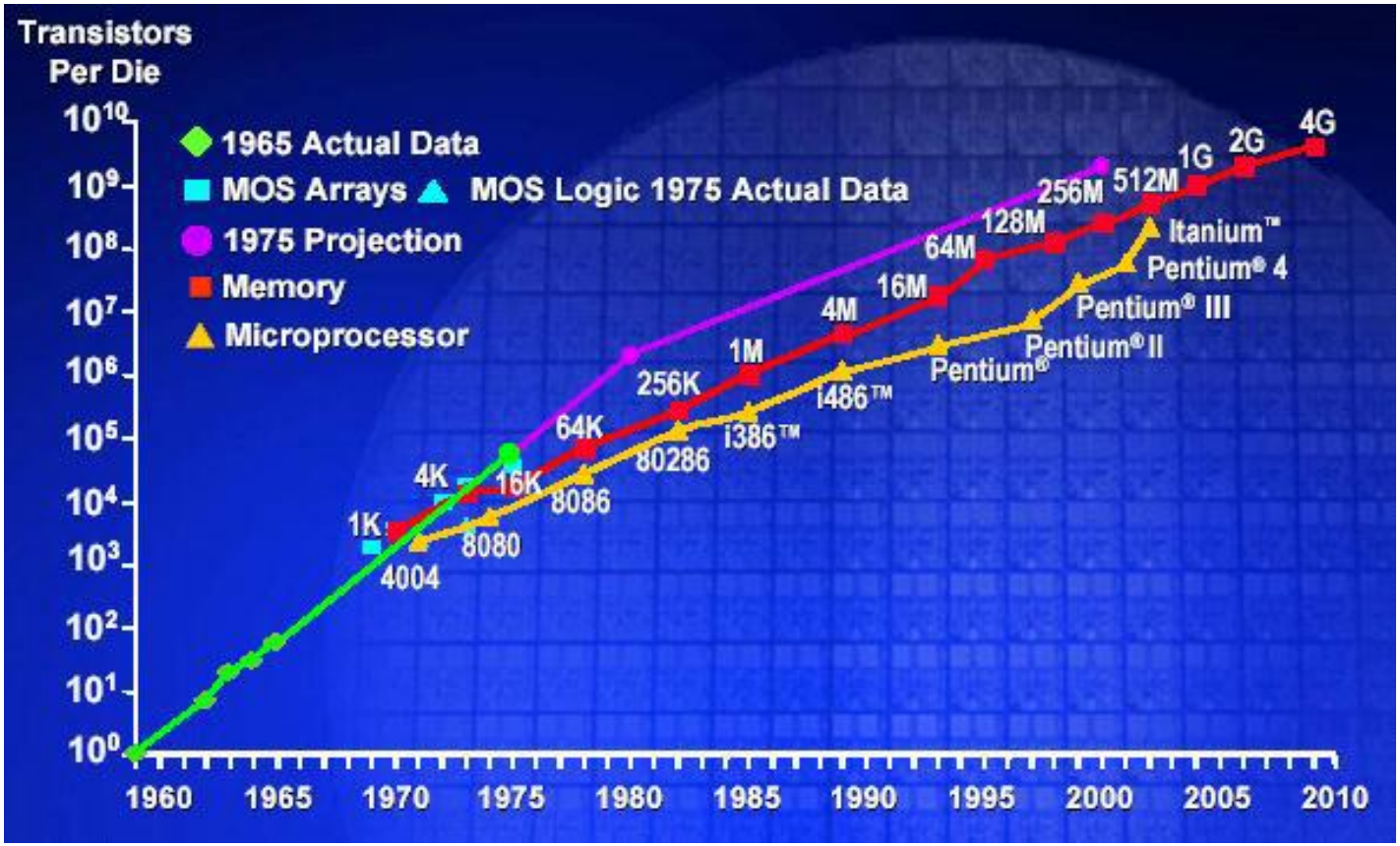
1.2.3 third generation – small- and medium-scale integrated circuit (IC) (1965-1971)

- era of microelectronics – invention of IC

- computer consists of :
  - gates (devices that implement simple Boolean function)
  - memory cells (devices that store one bit of data)
  - interconnections

Moore's Law

"The number of transistors incorporated in a chip will approximately double every 24 months." (1965)



1.2.4 later generations (1972-present)

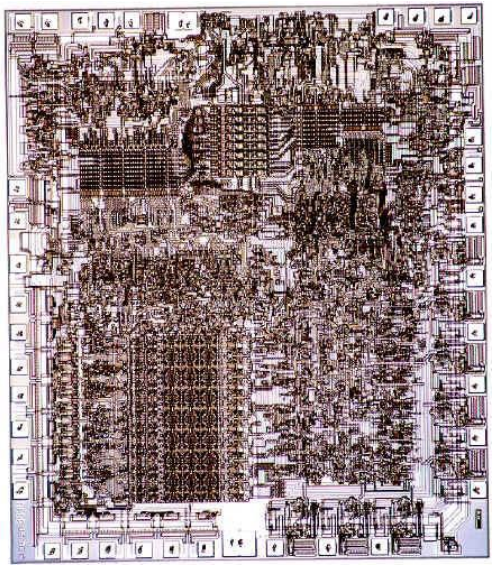
- introduction of large-scale integration (LSI, 1972-1977), very-large-scale integration (VLSI, 1978-1991), ultra-large-scale integration (ULSI, 1991 - ?)

- continuing increase in memory density, decrease in cost per bit, shorter access time

IBM Introduced its original PC in 1981

Used the Intel 8088 processor containing 29,000 transistors

Used operating system (MS-DOS) designed by Microsoft



Intel 8088

IBM PC Picture: Intel

Microprocessor Applications

NOT ONLY in conventional computers.

1. Smart phone
2. Mouse
3. Keyboard
4. Air conditioner
5. Fax machine
6. Fan
7. Toys

1.3 Review of number systems

- decimal and binary number systems
- converting between decimal to binary
- hexadecimal system
- converting between binary and hex
- addition, subtraction, multiplication, division
- unsigned and signed numbers
- BCD, ASCII

Number Representation

A positive number N can be written in positional notation as

$N = (a_{n-1}\ a_{n-2}\ \ldots\ a_1\ a_0\ )_r$

where $a_i \in (0,1, \ldots , r\text{-}1)$

r = radix or base of the number system being used

a_i = integer digit i when $n - 1 \geq i \geq 0$

a_{n-1} = most significant digit (MSD)

a_0 = least significant digit (LSD)

$$\textbf{decimal value} = a_{n-1}r^{n-1} + \ldots + a_1 r + a_0 r^0$$

1.3.1 decimal number

There are ten digits,(0-9), in this system.
decimal number, r = 10

From the Polynomial from:

$(a_{n-1}\ a_{n-2}\ \ldots\ a_1\ a_0)_r = a_{n-1}10^{n-1} + \ldots + a_0 10^0$

Example:

e.g. $(7392)_{10}$

$= 7\times10^3 + 3\times10^2 + 9\times10^1 + 2\times10^0$

1.3.2 binary number

A binary number system has only two digits, called bits. A binary number is a weighted number. The right-most bit is the least significant bit (LSB) and the left-most bit is the most significant bit (MSB).

Example: non negative integer

Decimal Number	Binary Number	
	MSB	LSB
0	0	0
1	0	1
2	1	0
3	1	1

In general, for non negative integer, with n bits, we can count up from 0 to 2^n - 1.

Binary-to-Decimal Conversion

From the Polynomial from:

$(a_{n-1}\, a_{n-2} \ldots a_1\, a_0)_r = a_{n-1}2^{n-1} + \ldots + a_0 2^0$

Example. Convert 0000 1101 to decimal form

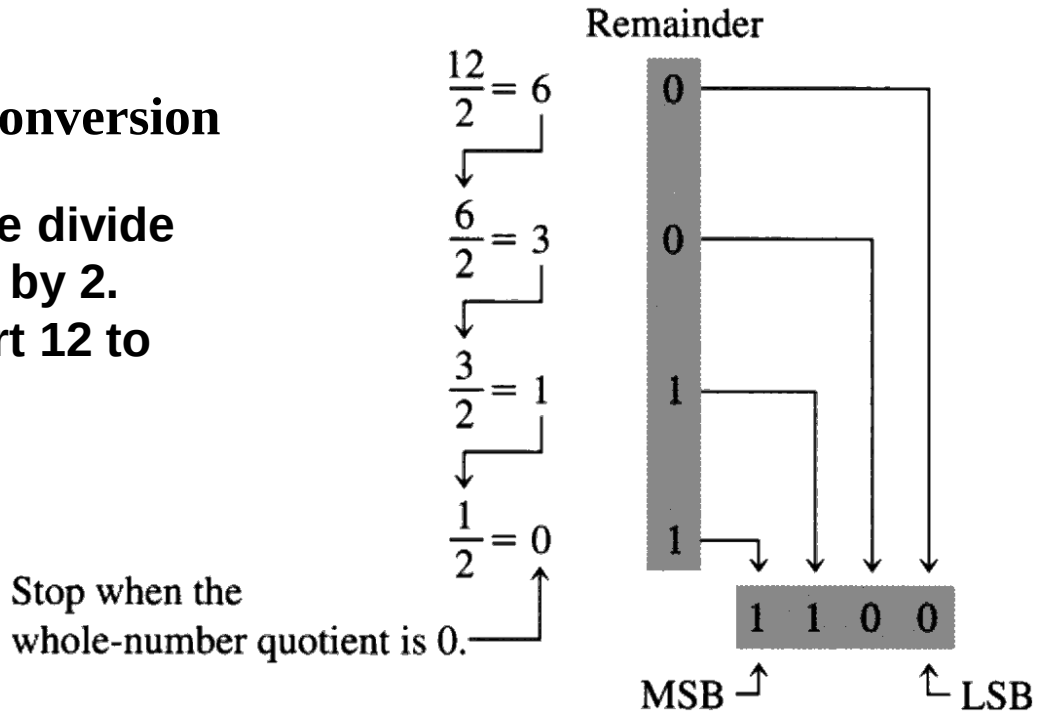
$0000\ 1101_2 = (1\text{x}2^3)+(1\text{x}2^2)+(1\text{x}2^0)$

$= 8 + 4 + 1 = 13$

0001 0101 = ?

Decimal-to-binary Conversion

Main step: Recursive divide the decimal number by 2. For example, convert 12 to binary number.



1.3.3 hexadecimal number

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Binary-to-Hexadecimal Conversion

Example: Convert 1100101001010111_2 to hexadecimal

Binary	1100	1010	0101	0111	
Hex.	C	A	5	7	$= CA57_{16}$

Hexadecimal-to-Decimal Conversion

- Convert the hexadecimal number to binary and then to convert from binary to decimal.
- Another way is to multiply the decimal value of each hexadecimal digit by its weight and then take the sum of these products.

Decimal-to-Hexadecimal Conversion

Repeated division-by-16 method.

Convert decimal number to hexadecimal

1128 = ?
1128/16=70, remainder=8
70/16=4, remainder=6
4/16=0, remainder=4
==> 468_{16}

256=?

DIVISION	RESULT	REMAINDER (in HEX)
256 / 16	16	0
16 / 16	1	0
1 / 16	0	1
ANSWER		100

590=?

DIVISION	RESULT	REMAINDER (HEX)
590 / 16	36	E (14 decimal)
36 / 16	2	4 (4 decimal)
2 / 16	0	2 (2 decimal)
ANSWER		24E

1.3.4 unsigned binary number

To represent non-negative number
Need to specify the number of bits used.

With n digits, 2^n unique numbers (from 0 to 2^n-1) can be represented.
If n=8, 256 (=2^8) numbers can be represented 0-255.

Convert 125 from decimal to

8 bit unsigned binary **Answer 0111 1101**

(NOT 111 1101)

Convert 96 from decimal to

8 bit unsigned binary **Answer 0110 0000**

(NOT 110 0000)

Convert 0001 0011 from binary to decimal **Answer 19**

Add the two 8-bit binary numbers (0011 1101) and (0001 0111).

$$
\begin{array}{cccccccc}
 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
+ & - & - & - & - & - & - & - & - \\
 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}
$$

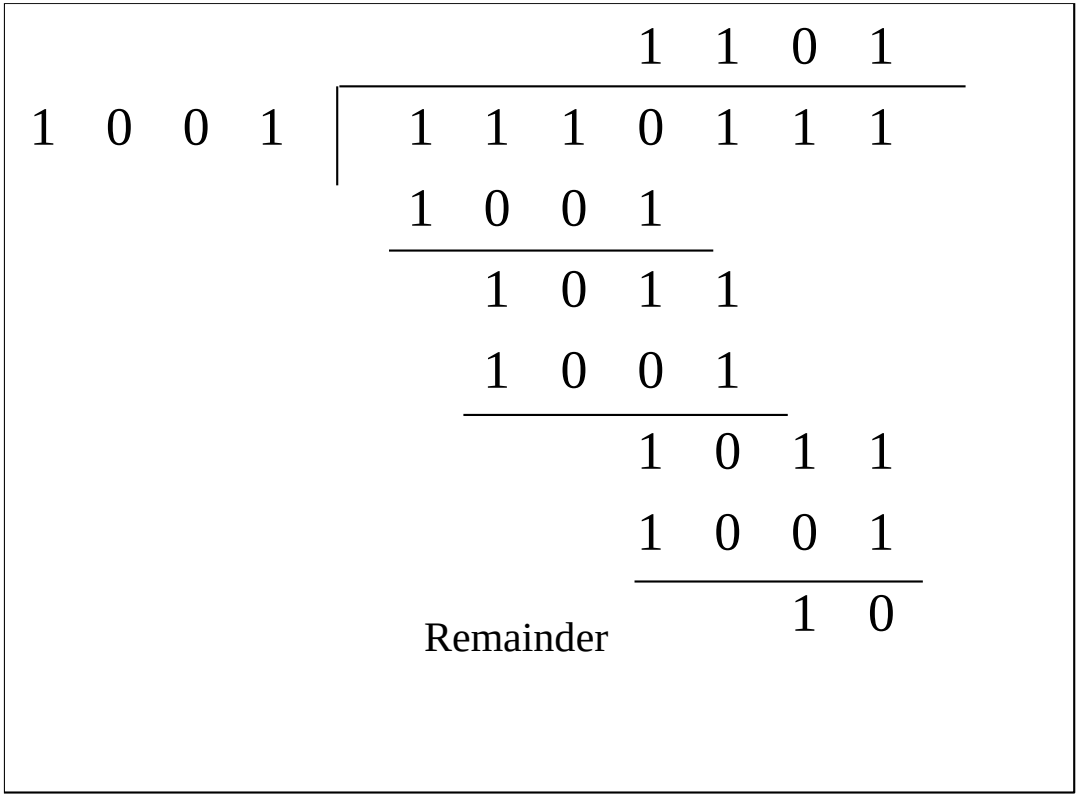
Subtract the two 8-bit binary numbers (0011 1101) and (0001 0111).

$$
\begin{array}{cccccccc}
 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
- & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
 & - & - & - & - & - & - & - & - \\
 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0
\end{array}
$$

Multiply two 8-bit binary numbers (0001 0111) and (0000 1010).

$$
\begin{array}{cccccccc}
 & & & 1 & 0 & 1 & 1 & 1 \\
\times & & & & 1 & 0 & 1 & 0 \\
\hline
 & & & 0 & 0 & 0 & 0 & 0 \\
 & & 1 & 0 & 1 & 1 & 1 & \\
 & 0 & 0 & 0 & 0 & 0 & & \\
1 & 0 & 1 & 1 & 1 & & & \\
\hline
1 & 1 & 1 & 0 & 0 & 1 & 1 & 0
\end{array}
$$

Division



1.3.5 signed binary number

In mathematics, negative numbers in any base are represented in the usual way, by prefixing them with a "−" sign.

However, in computer hardware, numbers are represented in bit, so a method of encoding the minus sign is necessary. Modern computers typically use the two's-complement representation, but other representations are used in some circumstances

1’s complement and 2’s complement

1's complement of a n-bit pattern is a transform that maps a n-bit pattern to another n-bit pattern. It simply changes all 1’s to 0’s and all 0’s to 1’s, as illustrated below:
Example, 1’s complement of a 8-bit pattern, 1011 0010=0100 1101

2's complement of a n-bit pattern is a transform that maps a n-bit pattern to another n-bit pattern
The 2's complement of a bit pattern is found by adding 1 to the LSB of the 1's complement.

2's complement = (1's complement) + 1 LSB bit

Example. Find the 2's complement of the binary number 010110010.

```
          0 1 0 1 1 0 0 1 0
          1 0 1 0 0 1 1 0 1   1's complement
+         0 0 0 0 0 0 0 0 1   Add 1 LSB bit
    ---------------------------
          1 0 1 0 0 1 1 1 0   2's complement
```

2's complement representation

positive number = same as the unsigned number,
negative number is the 2's complement of the corresponding positive number.

Example : express decimal numbers in the 8-bit 2's complement representation

25 = 0001 1001
-25 = 1110 0111

32 = 0010 0000

-128 =1000 0000
-1 =1111 1111
12 =0000 1100
-12 =1111 0100

From 2's complement to decimal: 1st bit = 0 -> +ve
1st bit = 1 -> -ve

If sign bit is 0, easy (same as "unsigned binary to decimal)
Example:
0000 1101 = $(1x2^3)+(1x2^2)+(1x2^0)$
= 8 + 4 + 1 = 13

If sign bit is 1, the magnitude of the negative number is equal to the decimal value of the 2's complement of the negative number.
Example
1110 0111=?
Sign bit =1 =>-ve
1110 0111 -> The magnitude 0001 1001 = 25=> -25
-> 1's complement then + 1

0011 0000=?
Sign bit =0=> +ve
48

Arithmetic Operations with 2's Complement Number System

Addition or Subtraction can be reduced to Addition

Addition
There are four cases:
Case 1: Both numbers are positive.

```
          0000 0111      7
+         0000 0100      4
-------------------------
          0000 1011      11
```

Case 2: Positive number larger than negative number.

```
          00001111      15
+         11111010     -6
-------------------------  ---
        1 00001001      9
```

Discard carry

No need to use the carry bit to check overflow!
We use other method

Arithmetic Operations with 2's Complement Number System

Case 3: Negative number larger than positive number.

```
            00010000        16
+           11101000       -24
-------------------------  ----
            11111000        - 8
```

The sum is negative and therefore is in 2's complement form.

Case 4: Both numbers are negative.

```
            11111011        - 5
+           11110111        - 9
-------------------------
        1   11110010       -14
```

Arithmetic Operations with 2's Complement Number System

Overflow Condition

When two signed numbers are added and the number of bits required to represent the sum exceeds the number of bits in the two numbers, an overflow result is indicated by an incorrect sign bit.

An overflow can occur only when both numbers are positive or negative. 125+58

```
          01111101
+         00111010
--------------------------
          10110111
```

From the binary calculation, +ve + +ve = -ve => impossible => Overflow.

Check sign bits.

1.3.6 BCD code

Binary Coded Decimal (BCD) is a way to express each of the decimal digits with a binary code. Since there are only ten code groups in the BCD system, it is very easy to convert between decimal and BCD. Because we like to read and write in decimal, the BCD code provides an excellent interface to binary systems

8421 Code

BCD	decimal	BCD	decimal
0000	0	0101	5
0001	1	0110	6
0010	2	0111	7
0011	3	1000	8
0100	4	1001	9

16 = 0001 0110
15 = 0001 0101

Rules to perform BCD addition:

Add in Binary but

(i) add 110 to the result if it is between 1010 and 1111

(ii) add 110 to the result if there is a carry

Example. Add the following two 2-digit BCD numbers:

00010110 + 00010101

```
 0001   0110
+0001   0101
----------------------
 0010   1011     Right group is invalid (>9), left group
                 is valid. Add 6 to invalid code
+
          0110
----------------------
 0011   0001     = 31
```

1.3.7 ASCII code

The most widely used character code in computer applications is the *ASCII* (American Standard Code for Information Interchange) code.

Example
Encode the word Boy in ASCII code, representing each character by two hexadecimal digits.

Character	Binary Code	Hexadecimal Code
B	0100 0010	42
o	0110 1111	6F
y	0111 1001	79

Example

Determine the decimal value of the number 11101000.

Ans: ???? (We do not know because we do not know the format)

Now, if we know the format :
if it is a binary number of

1. **8-bit unsigned system representation**
2. **8-bit 2's complement system representation**
3. **BCD representation**

1) **232**
2) **-24**
3) **invalid because 1110 = ? In BCD**

Example

Determine the decimal value of the number 0100 0001.

if it is a binary number of
1. 8-bit unsigned system representation
2. 8-bit 2's complement system representation
3. BCD representation

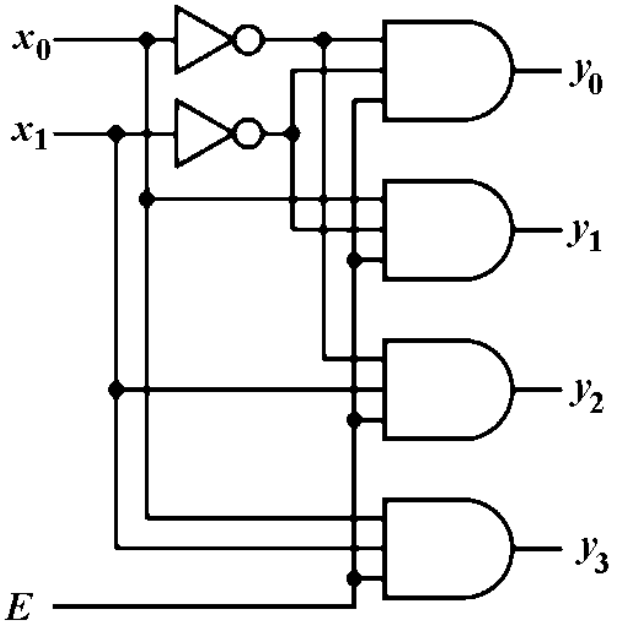
1) 65
2) 65
3) 41

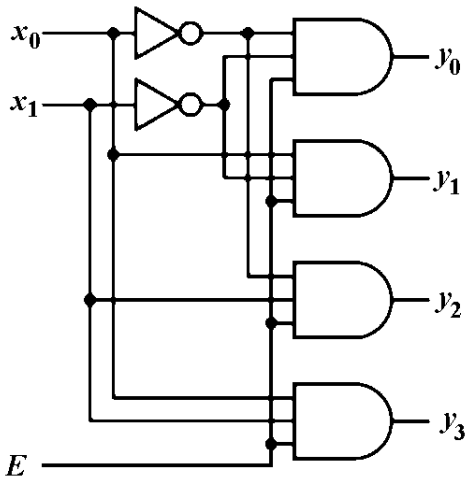
1.4 Review basic logic circuits

- logic gates - AND, OR, NOT, XOR, NAND, NOR
- Boolean algebra
- flip-flops

Decoder

A decoder can take the form of a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different. e.g. n-to-2^n, BCD decoders.





E	x_1x_0	y_3	y_2	y_1	y_0
0	XX	0	0	0	0
1	00	0	0	0	1
1	01	0	0	1	0
1	10	0	1	0	0
1	11	1	0	0	0

Decoding is necessary in applications such as data multiplexing, 7 segment display and memory address decoding.

1.5 Structure of computer

Important terminology:

Bit	0	
Nibble	0000	(4 bits)
Byte	0000 0000	(8 bits)
Word	0000 0000 0000 0000 0000 0000 0000 0000	(4 bytes, 32 bits)

KB (kilobyte) = 2^{10} bytes

MB (megabyte) = 2^{20} bytes

GB (gigabyte) = 2^{30} bytes

TB (terabyte) = 2^{40} bytes

1.5.1 basic computer architecture

Computer:

Perform a number of elementary computer operations (**instructions**) that manipulate information, called **data**.

Instruction Set:

The collection of all the instructions is called the instruction set of that computer.

Program:

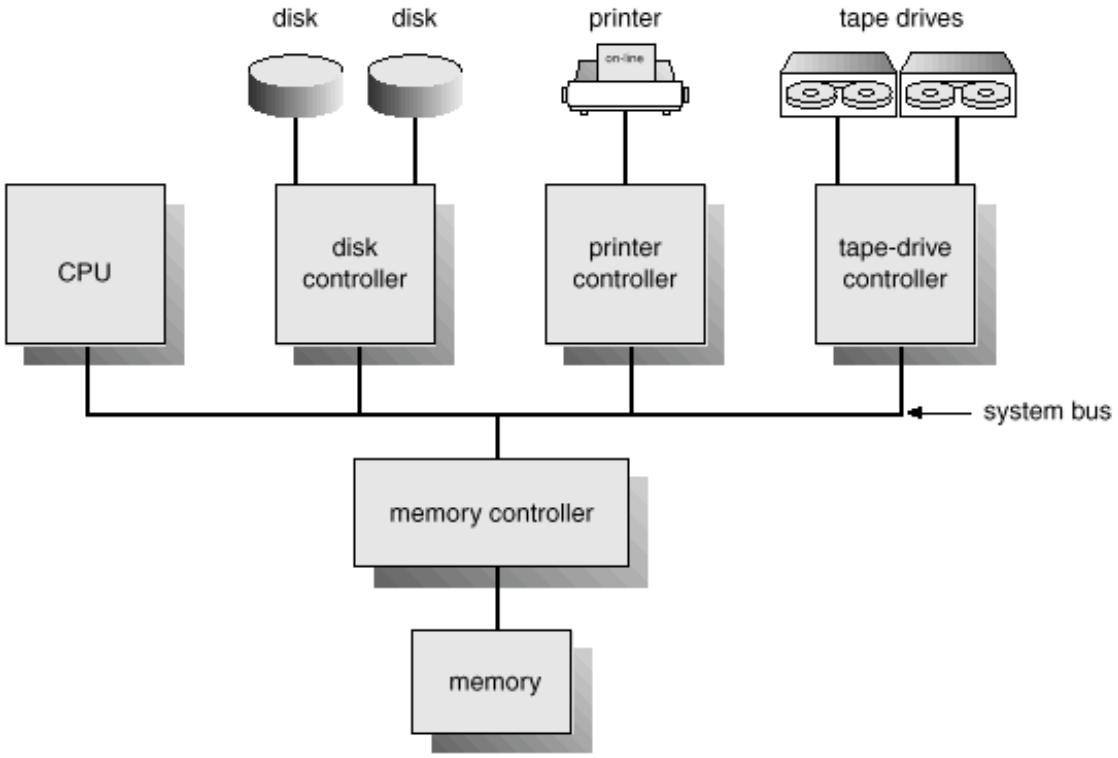
Consists of a number of instructions.

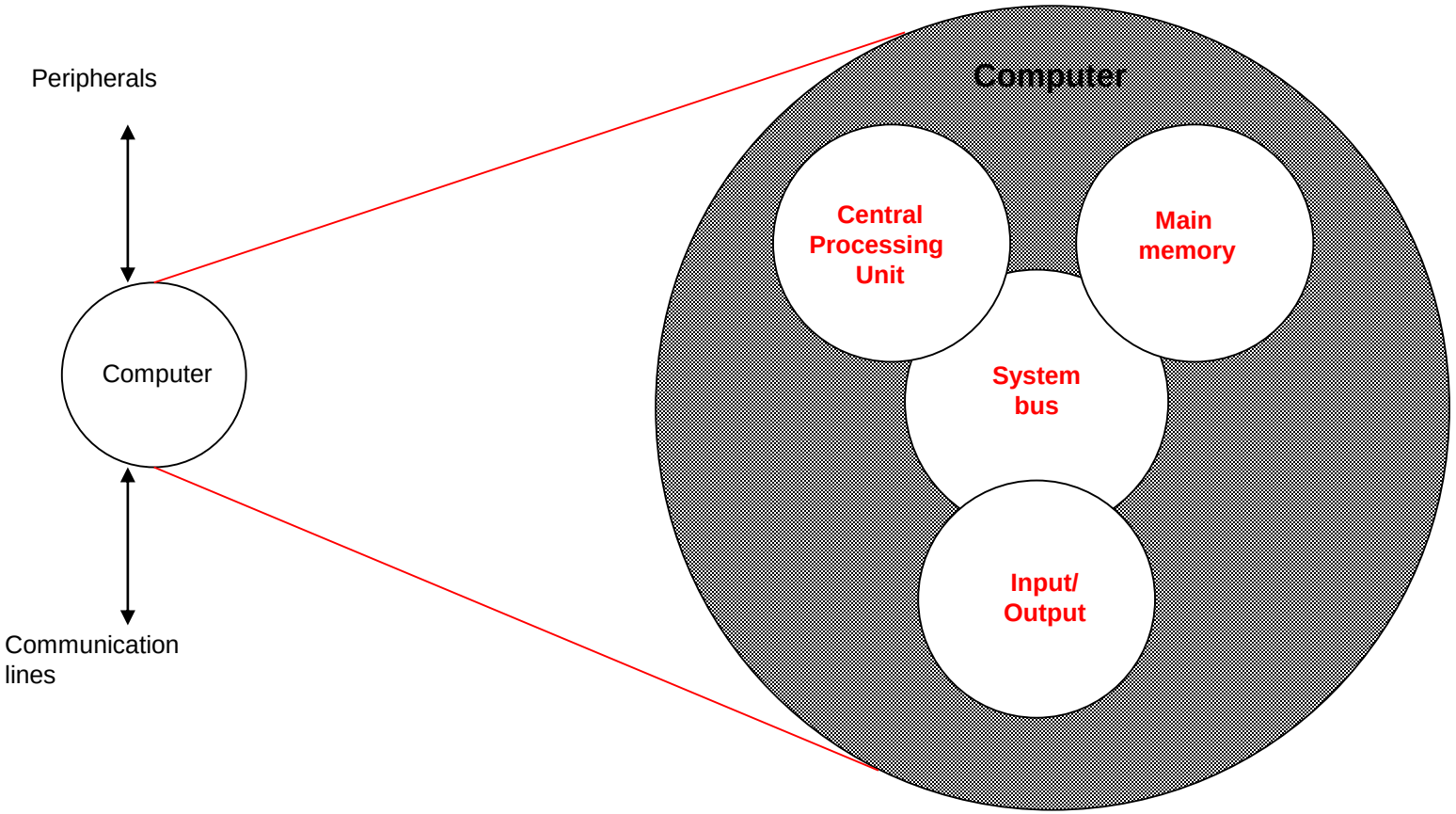
A digital computer may be divided into three different parts:

Central Processing Unit (CPU)

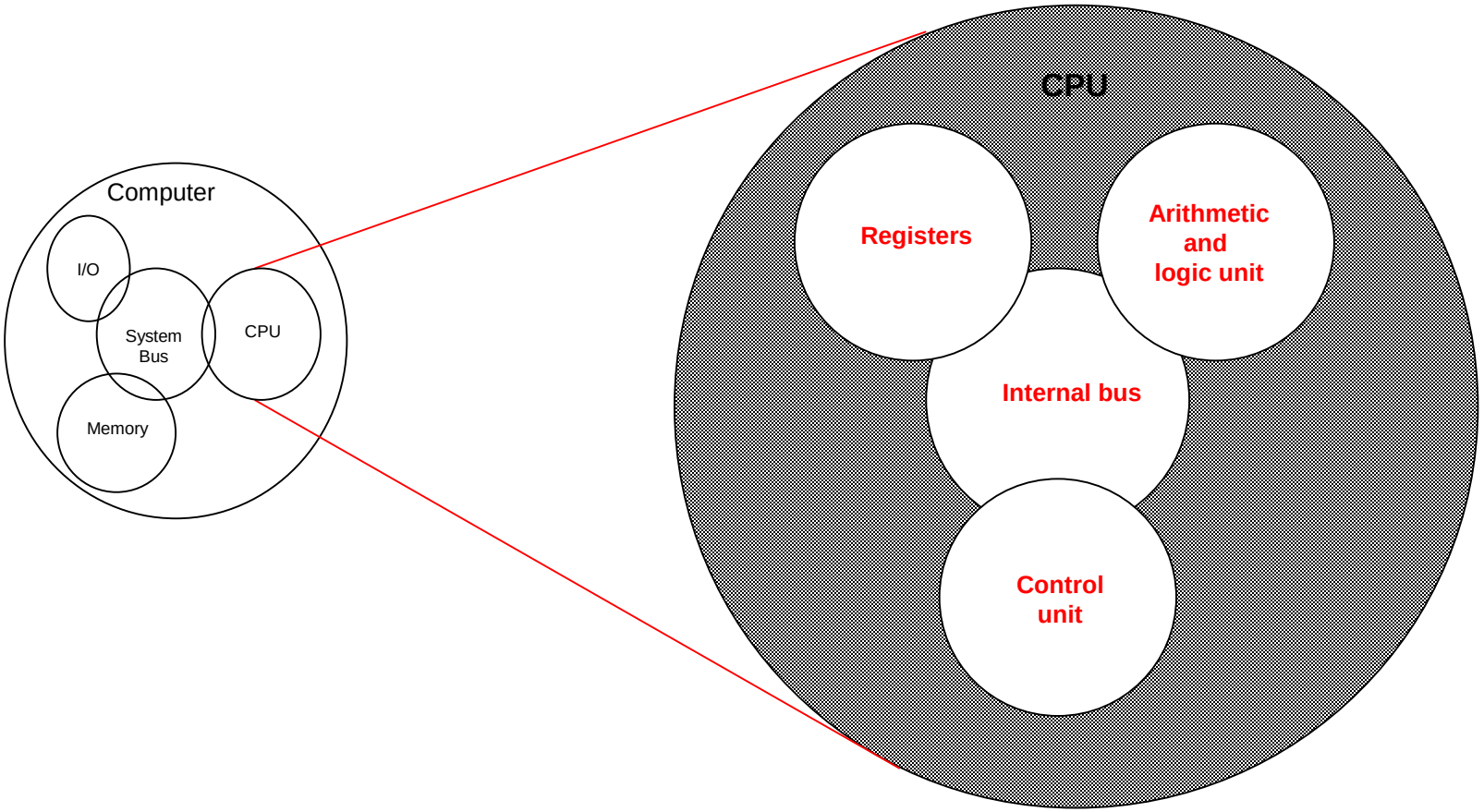
Memory

Input/Output Devices (I/O)

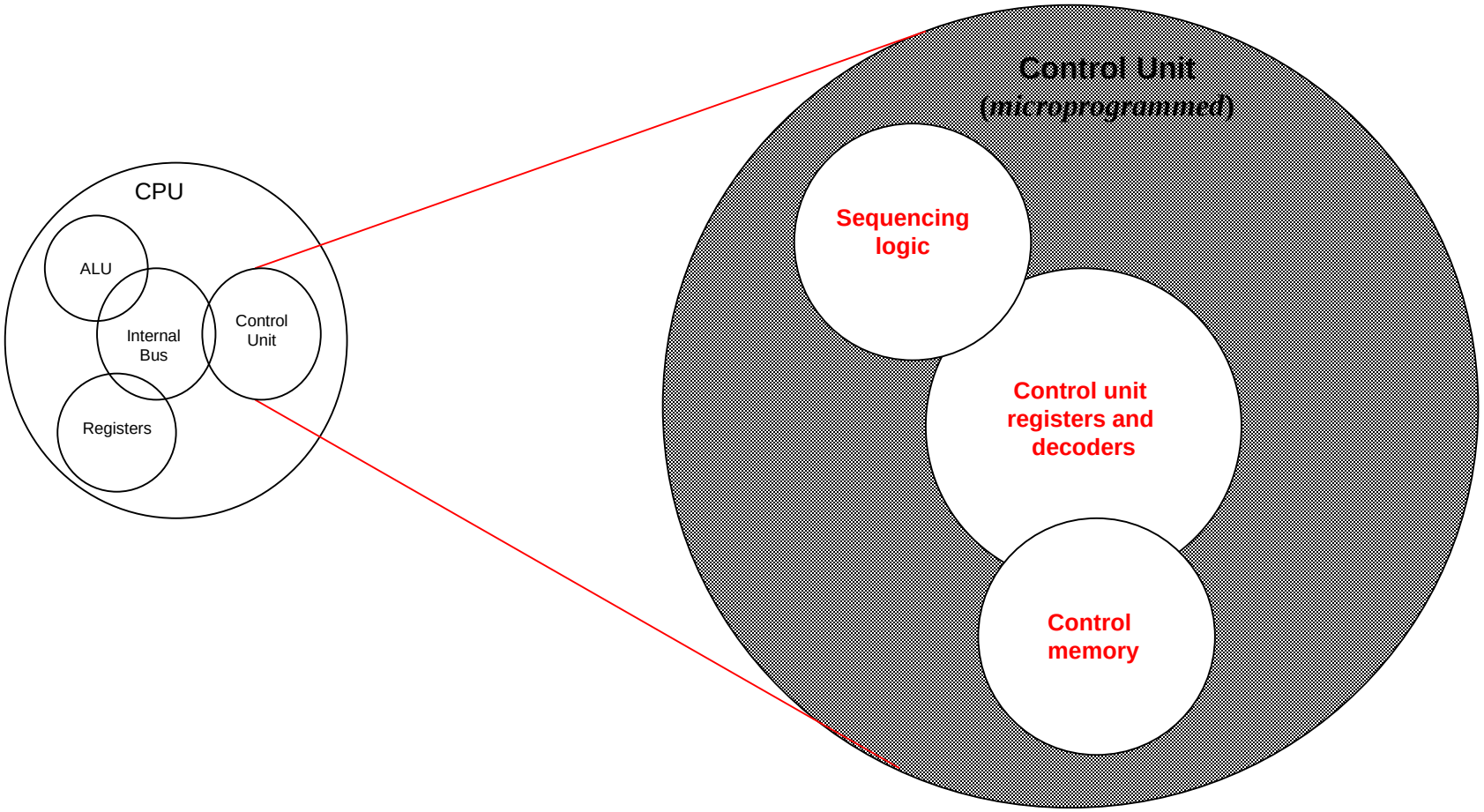




top-level structure with 4 main structural components



next level with 4 structural components

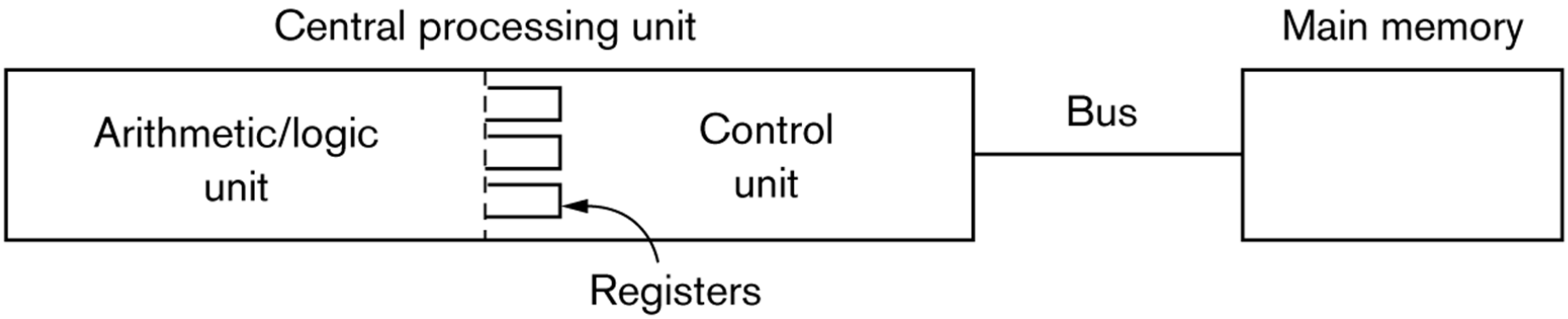


lowest level

1.5.2 CPU

Inside a CPU, there are

- Arithmetic and Logic Unit (ALU): to perform the arithmetic and logic operations
- Control unit : coordinating the machine's activities
- Registers : to store temporary data



Arithmetic and Logic Unit (ALU), **and Control Unit**

The ALU performs all the numerical computations and logical evaluations for the processor. The ALU receives data from the memory, performs the operations, and, if necessary, writes the result back to the memory.

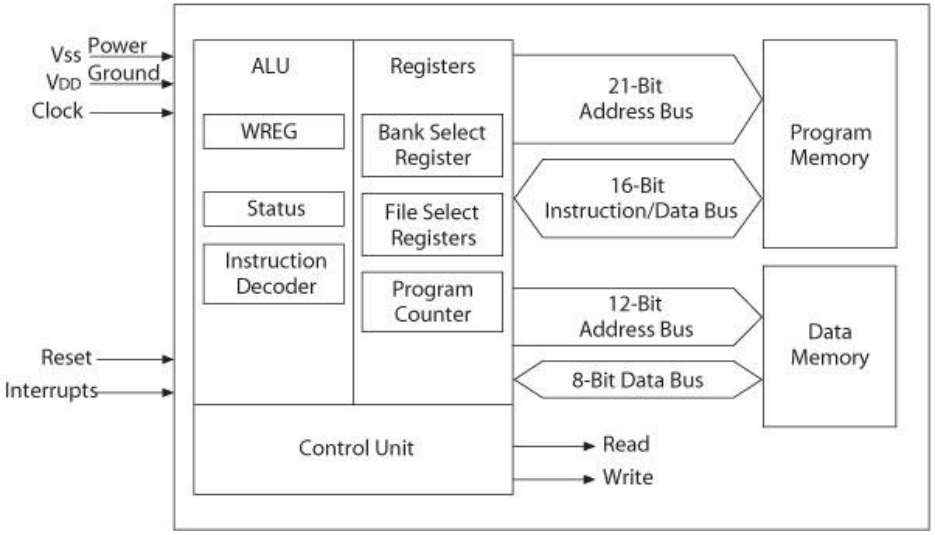
The control unit contains the hardware instruction logic. The control unit decodes and monitors the execution of instructions. The control unit also acts as an arbiter as various portions of the computer system compete for the resources of the CPU. It coordinates the operations of I/O units, memory, and ALU. It is effectively the nerve center that sends control signals to other units and senses their states.

Remark: In order to access a data item or an instruction, CPU needs to know their addresses. How ? Where: Program Counter (PC)

- Accumulator and General Purpose Registers
  - Accumulator (WREG)
  - Registers
- Special-purpose registers
  - Program Counter (PC)
  - Instruction Register (IR)
  - Stack Pointer
  - Status Word Register
  - In PIC18, there are Bank and File Select Registers

    Bank Select Register: select which data space in RAM is active

    File Select Register: for indirect addressing



- Stack Pointer permits “context switching” in interrupt service routines (ISR) and subroutines

Accumulator (WREG)

A register stores the results of arithmetic and logic operation.

Data Registers and Address Registers.

Registers stores data or addresses.

Program Counter (PC)

Store the address of the instruction to be executed in the next instruction cycle.

Is updated automatically.

Instruction Register (IR)

Store the instruction currently being executed or decoded

Stack Pointer

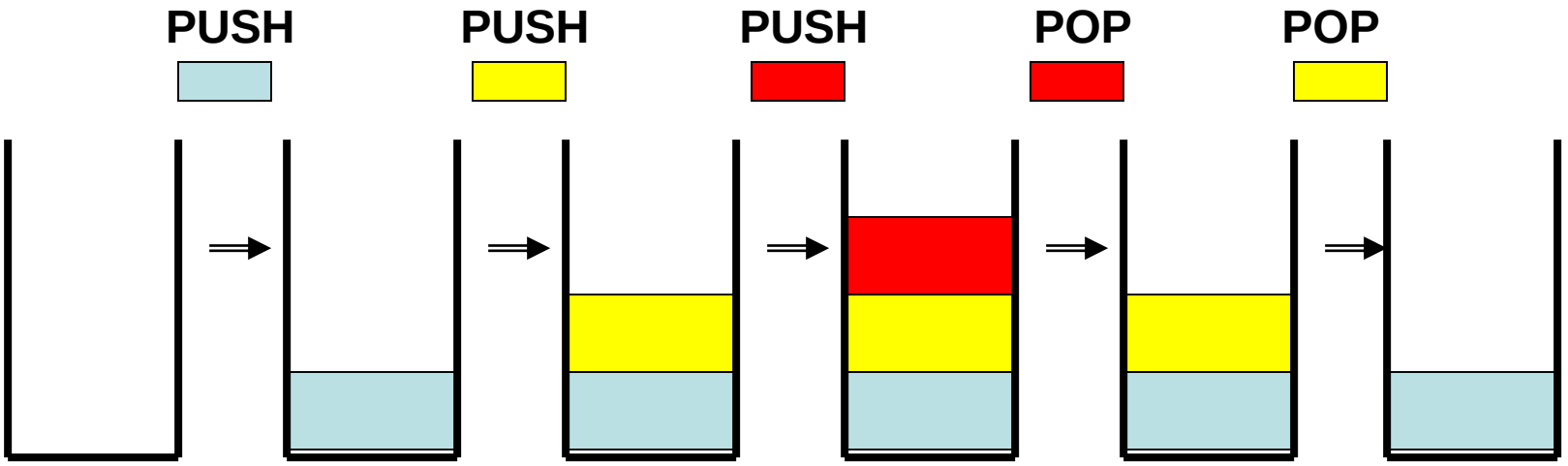
- Help the implementation of a stack data structure.

Status Word Register

- Contain several bits.
- Each bit indicates the state of CPU.
- For example, a carry flag indicates if the last addition operation produces a carry or not.

Stack

- Stack：a section of RAM to store data items
- Stack register (stack pointer): point to the location of the top of the stack.
- Two operations on the stack：
  - PUSH：put an item onto the *top* of the stack
  - POP：remove an item from the *top* of the stack



1.5.3 memory, I/O

Memory: Store programs and data.

A program contains many instructions stored in memory.

Each memory unit or location has **an address**.

Primary storage and secondary storage

Primary (IC chips in the computer) – to hold data and programs which are currently used.

Secondary (disk) – to hold large amount of data or programs which are not currently used.

In some computer systems, program and data share the same memory space.

In some computer systems, program and data are stored in separate memory spaces.

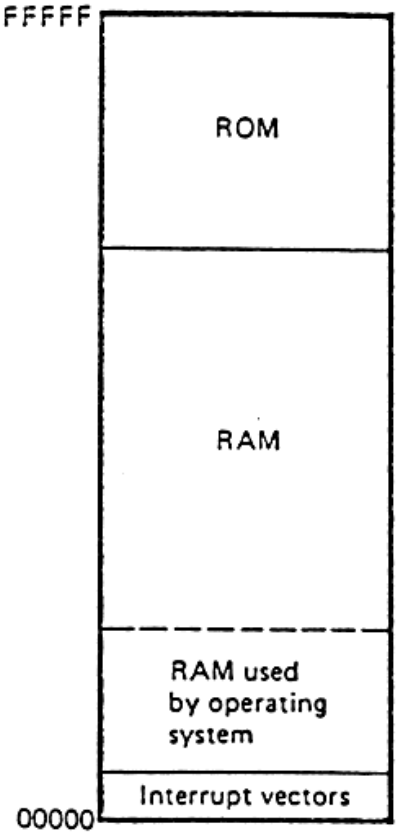
Input units: Computer accepts information through input units, which read the data. (e.g. keyboard, mouse). Each input unit has a corresponding address.

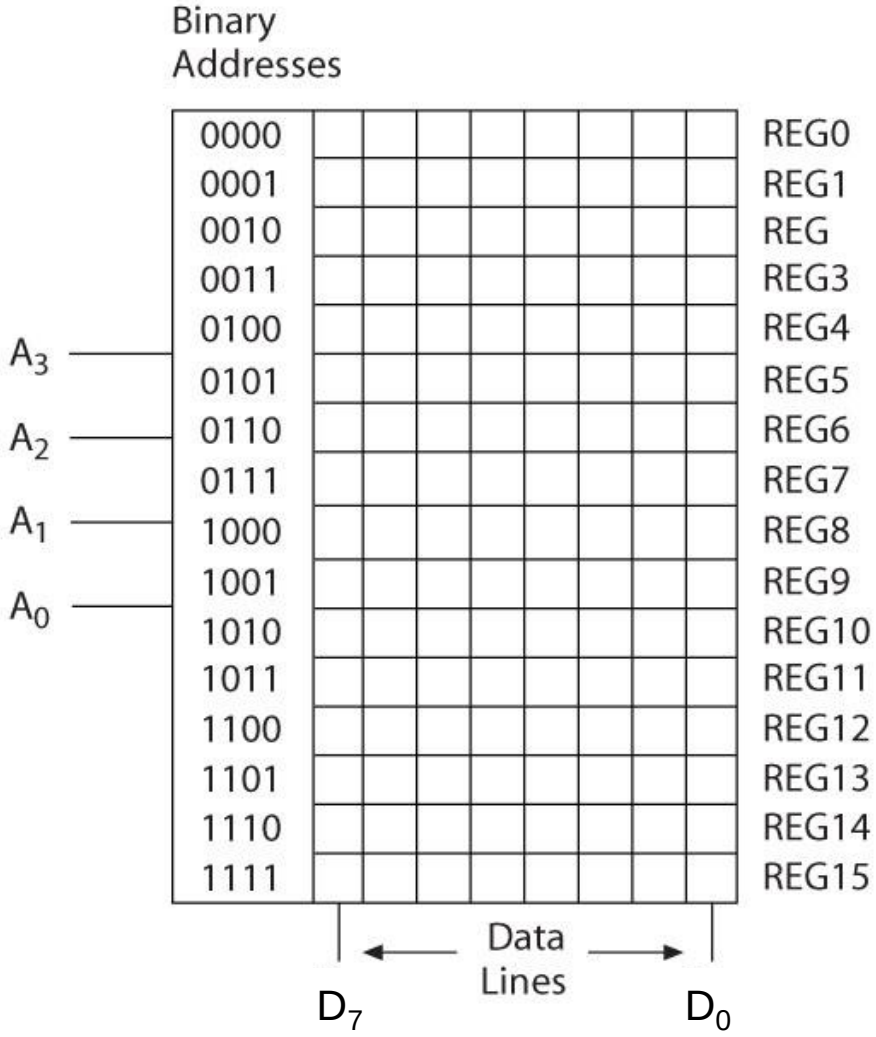
Output units: Counterpart of the input unit. Send processed results to the outside world. (e.g. monitor). Each output unit has a corresponding address.

Memory

The memory is used to store programs and data. The memory is arranged sequentially into a number of units, each is capable of holding one computer word. Each unit is referred to as a memory location and identified by a unique address. The binary word stored in a particular location is referred to as the content of that location.

Binary Address	Memory Content 8 bits
	:
:	:
:	:
0100	:
0011	content of address 0011
0010	content of address 0010
0001	content of address 0001
0000	content of address 0000

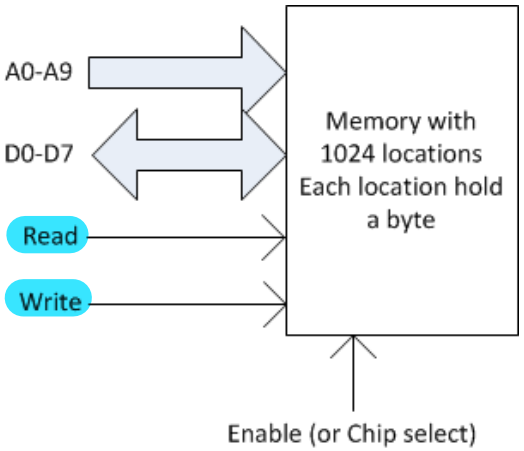




- A semiconductor storage device consisting of registers that store binary bits
- Two major categories
  - Read/Write Memory (R/W)
  - Read-Only-Memory (ROM)

ROM vs RAM

- ROM (read only memory)
  - contains programs and information essential to operation of the computer
  - for permanent data which cannot be changed by the user
  - nonvolatile, data does not lost when power off
- RAM (random access memory)
  - for temporary storage of programs that it is running
  - volatile, data lost when power off



Address lines: Wires carry an address of a location being accessed.

(unidirectional from external circuit to memory)

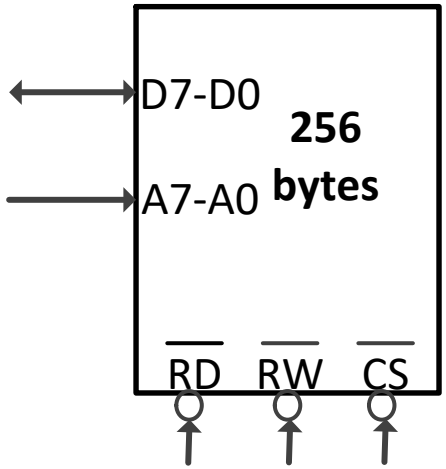
Data lines: Wires carry the data content being transferred between external circuit and memory (bidirectional)

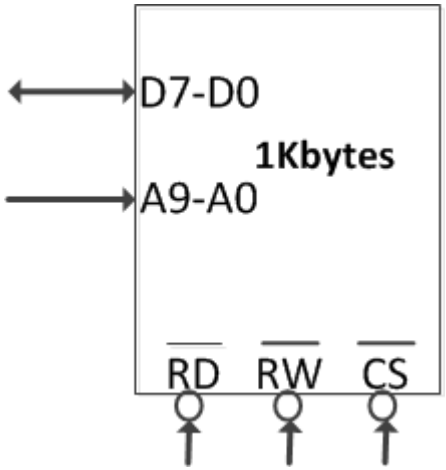
Enable: a wire carries control signal to enable the memory chip.

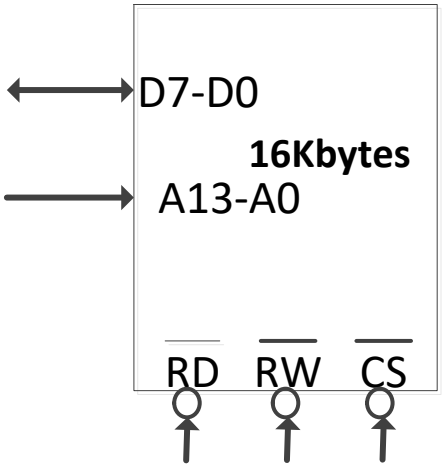
(unidirectional from external circuit to memory)

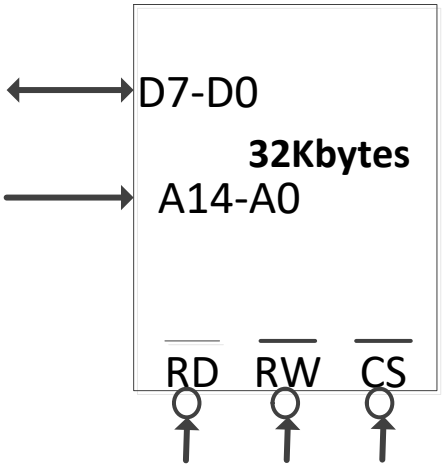
Read: a wire carries control signal to instruct the memory chip the operation is a read operation. (unidirectional from external circuit to memory)

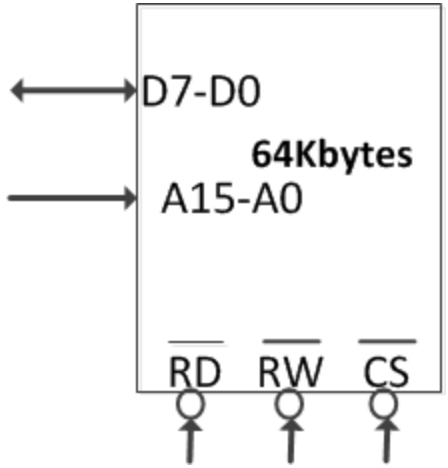
Write: a wire carries control signal to instruct the memory chip the operation is a write operation. (unidirectional from external circuit to memory)

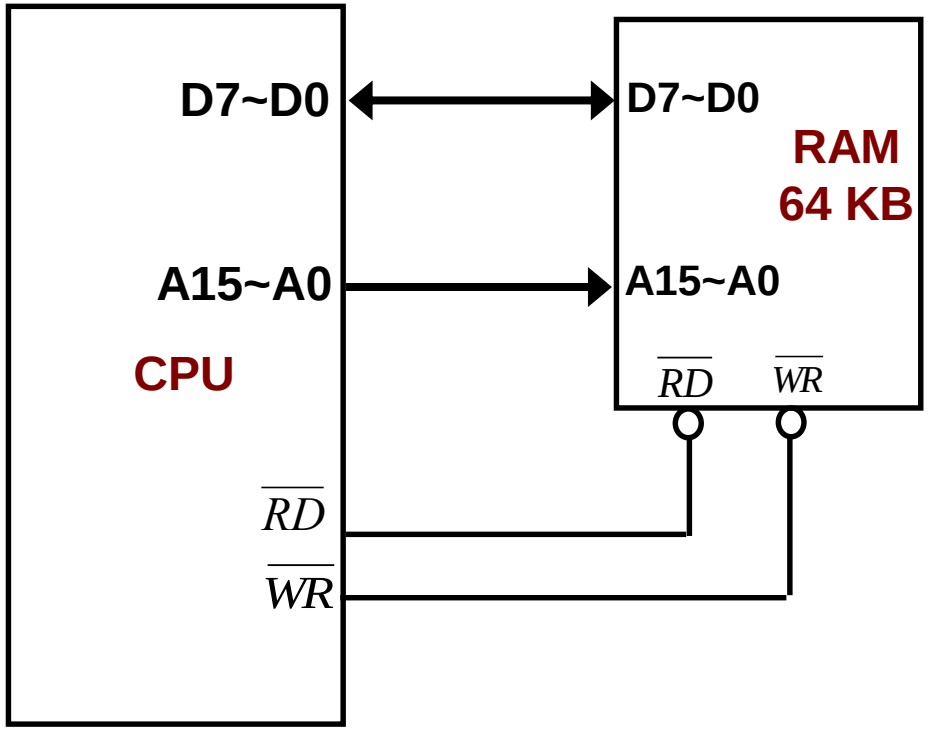

D7-D0
A7-A0
256
bytes
RD
RW
CS


D7-D0
A9-A0
1Kbytes
RD
RW
CS

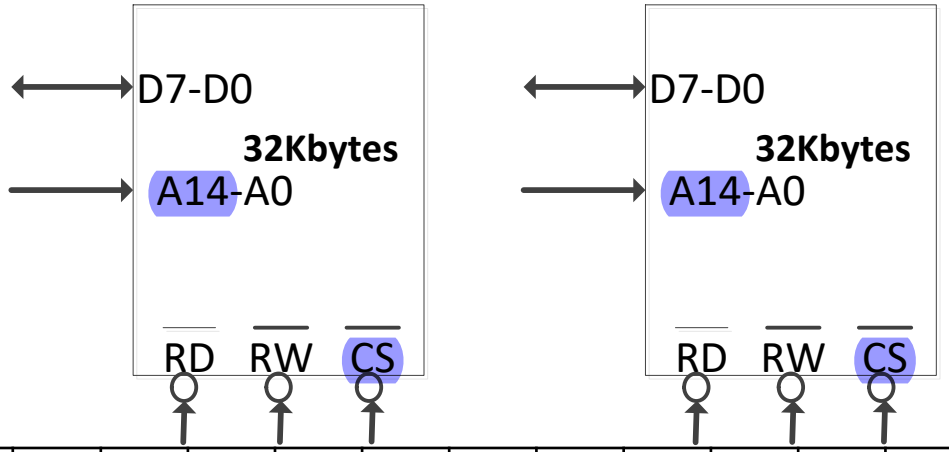

D7-D0
A13-A0
16Kbytes
RD
RW
CS


D7-D0
A14-A0
32Kbytes
RD
RW
CS


D7-D0
A15-A0
64Kbytes
RD
RW
CS

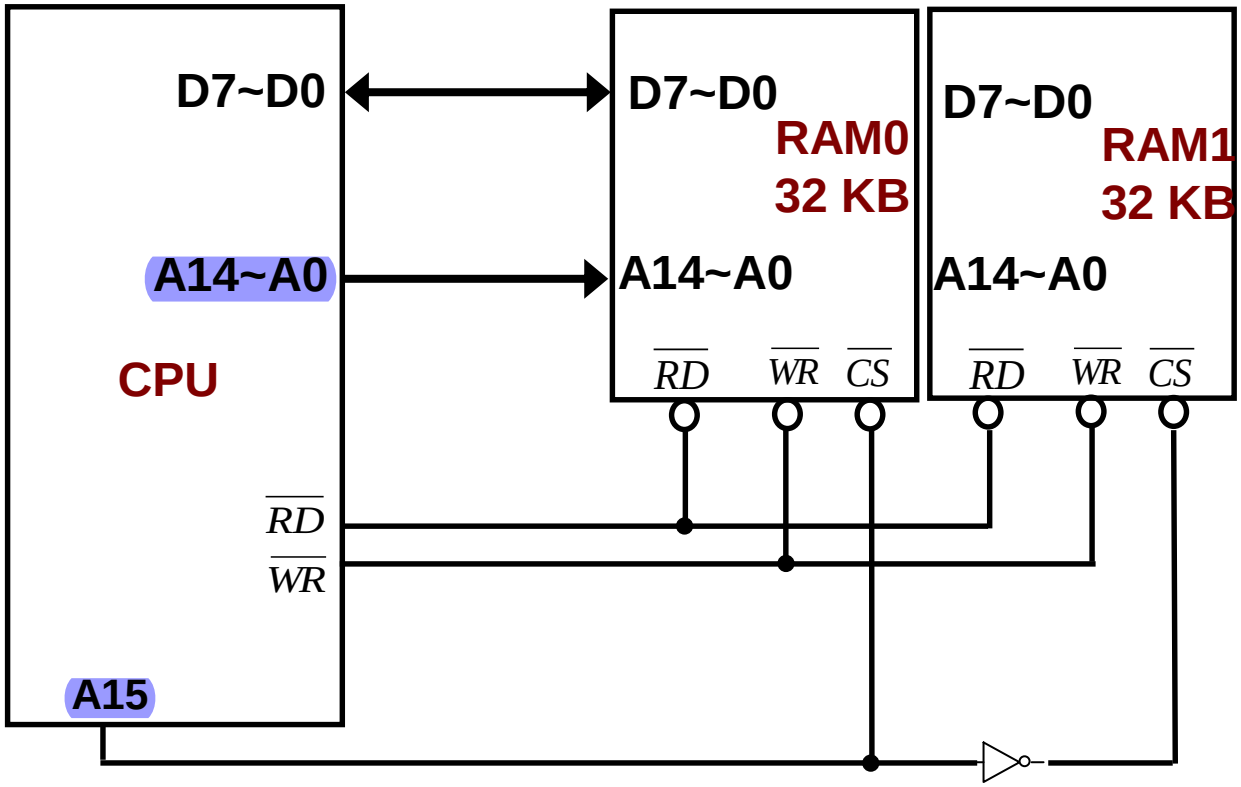

D7~D0
A15~A0
CPU
$\overline{RD}$
$\overline{WR}$
D7~D0
RAM
64 KB
A15~A0
$\overline{RD}$
$\overline{WR}$

- CPU has 64 KB memory
- Two memory chips. Each is 32 KB
- Two RAM area is from 0000h to 7FFFh (RAM0) and 8000h to FFFFh (RAM1)



	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RAM0 0000 to 7FFF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM1 8000 to FFFF	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

- There are two 32 KB RAMs
- A15 applied to select one RAM chip
- Two RAM area is from 0000h to 7FFFh (RAM0)
  and 8000h to FFFFh (RAM1)



Address Decoding

- Given an n-bit address, build a circuit to detect a range of addresses
- Example: Given a 16-bit address (A15-A0),
  design a decoder to detect 0000h to 7FFFh
  (**assume active low**. When the address is in the range, the output of decoding circuit is 0.)

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1000 to 7FFF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

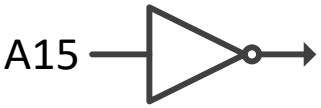
=> **Use A15**

Address Decoding

- Example: Given a 16-bit address (A15-A0), design a decoder to detect 8000h to FFFFh (**assume active low**. When the address is in the range, the output of decoding circuit is 0.)

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
8000 to FFFF	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

=> **Use $\overline{\text{A15}}$**

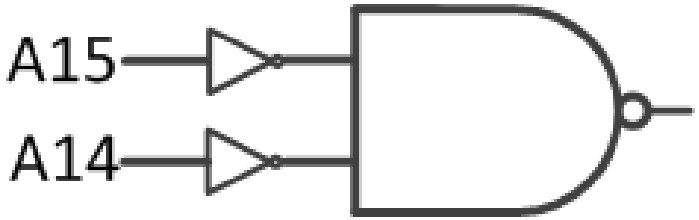


Address Decoding

- Example: Given a 16-bit address (A15-A0), design a decoder to detect 0000h to 3FFFh (**assume active low**. When the address is in the range, the output of decoding circuit is 0.)

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0000-3FFF	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x

=> Use $\overline{\mathbf{A15}}$ and $\overline{\mathbf{A14}}$

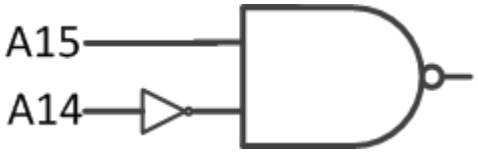


Address Decoding

- Example: Given a 16-bit address (A15-A0), design a decoder to detect 8000h to BFFFh (**assume active low**. When the address is in the range, the output of decoding circuit is 0.)

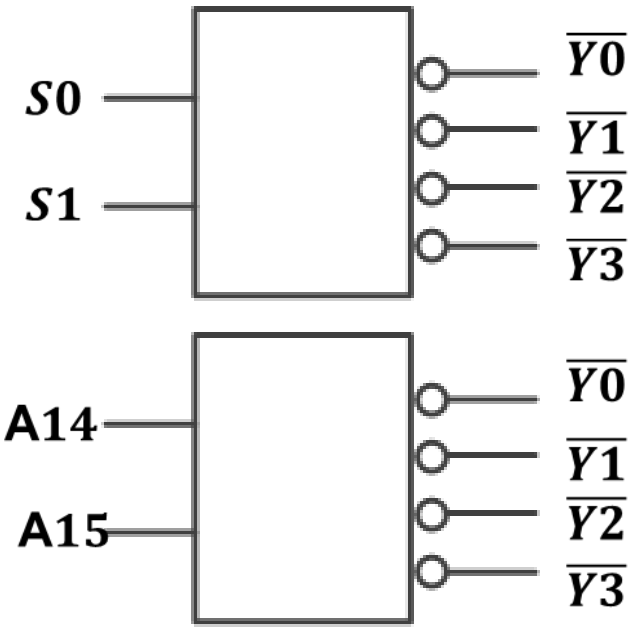
	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
8000-BFFF	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x

=> Use A15 and $\overline{\text{A14}}$



Address Decoding

- Use decoder



S1	S0	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

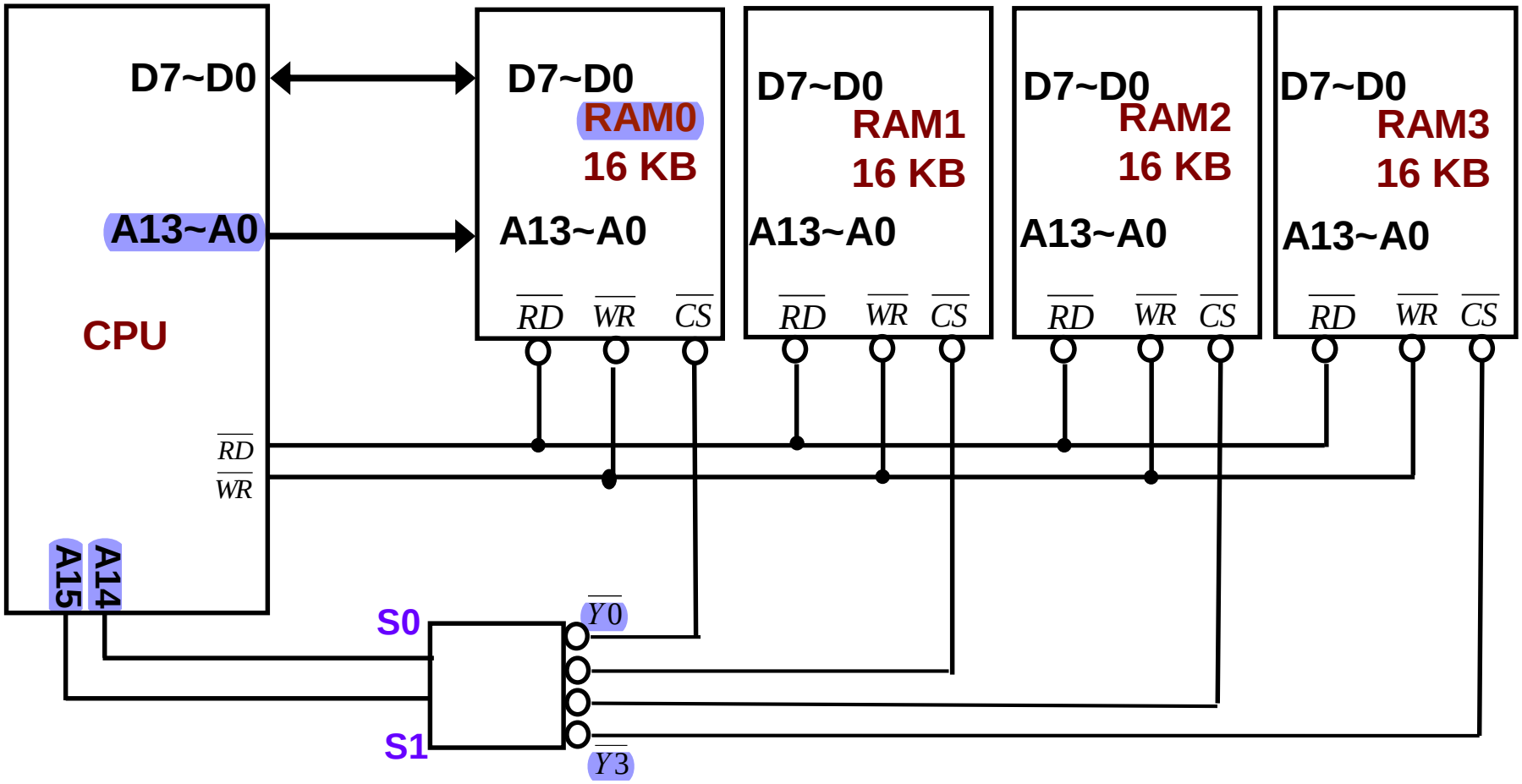
$\overline{Y0}$ detects A15 A14 = 0 0, Address range 0000h – 3FFFh

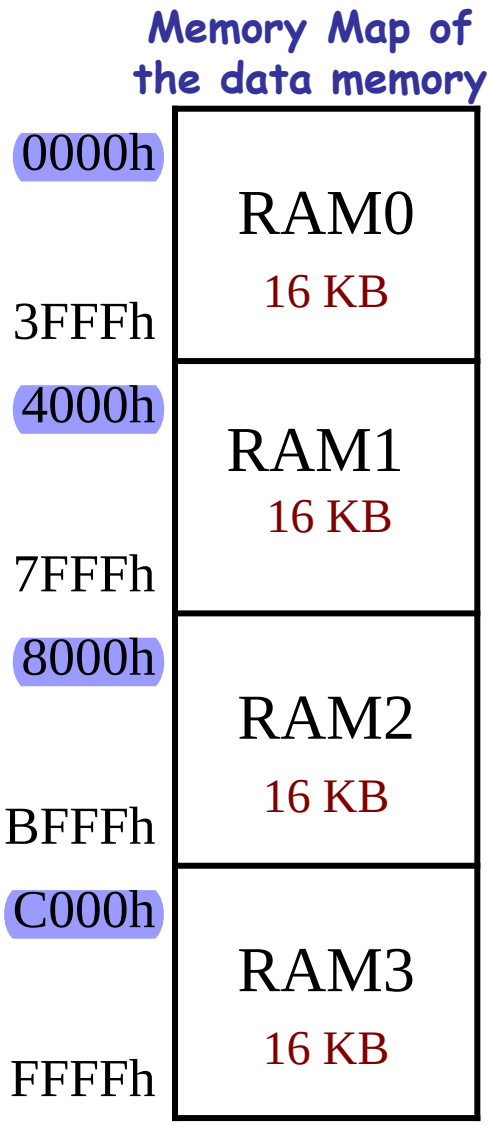
$\overline{Y1}$ detects A15 A14 = 0 1, Address range ?? 4000h - 7FFFh

$\overline{Y2}$ detects A15 A14 = 1 0, Address range ?? 8000h - BFFFh

$\overline{Y3}$ detects A15 A14 = 1 1, Address range ?? C000h - FFFFh

- There are four 16 KB RAMs
- Use A14 and A15 to select one RAM chip



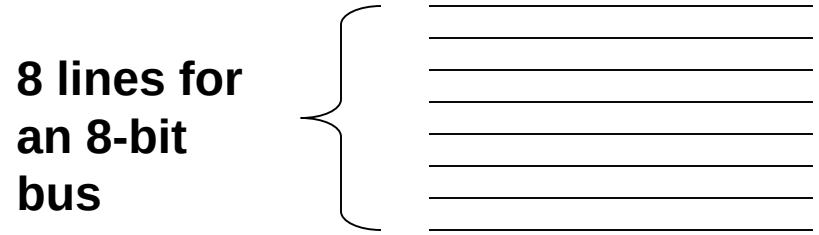

Memory Map of
the data memory
0000h
RAM0
16 KB
3FFFh
4000h
RAM1
16 KB
7FFFh
8000h
RAM2
16 KB
BFFFh
C000h
RAM3
16 KB
FFFFh

1.5.4 system bus

- CPU is connected to memory and I/O through strips of wire called system bus.
- Buses are used to communicate between the computer components.
  - Data Bus
  - Address Bus
  - Control Bus

Data Bus

- Size of data bus is larger, the better the CPU is.
  - Example：8 bits（slow）, 16 bits, 32 bits, 64 bits（fast）.
  - An 8-bit data bus can send 1 byte a time.
- Data bus is bi-directional.
- Larger data bus means a more expensive CPU and computer.
- The processing power of a computer is related to the size of its buses.



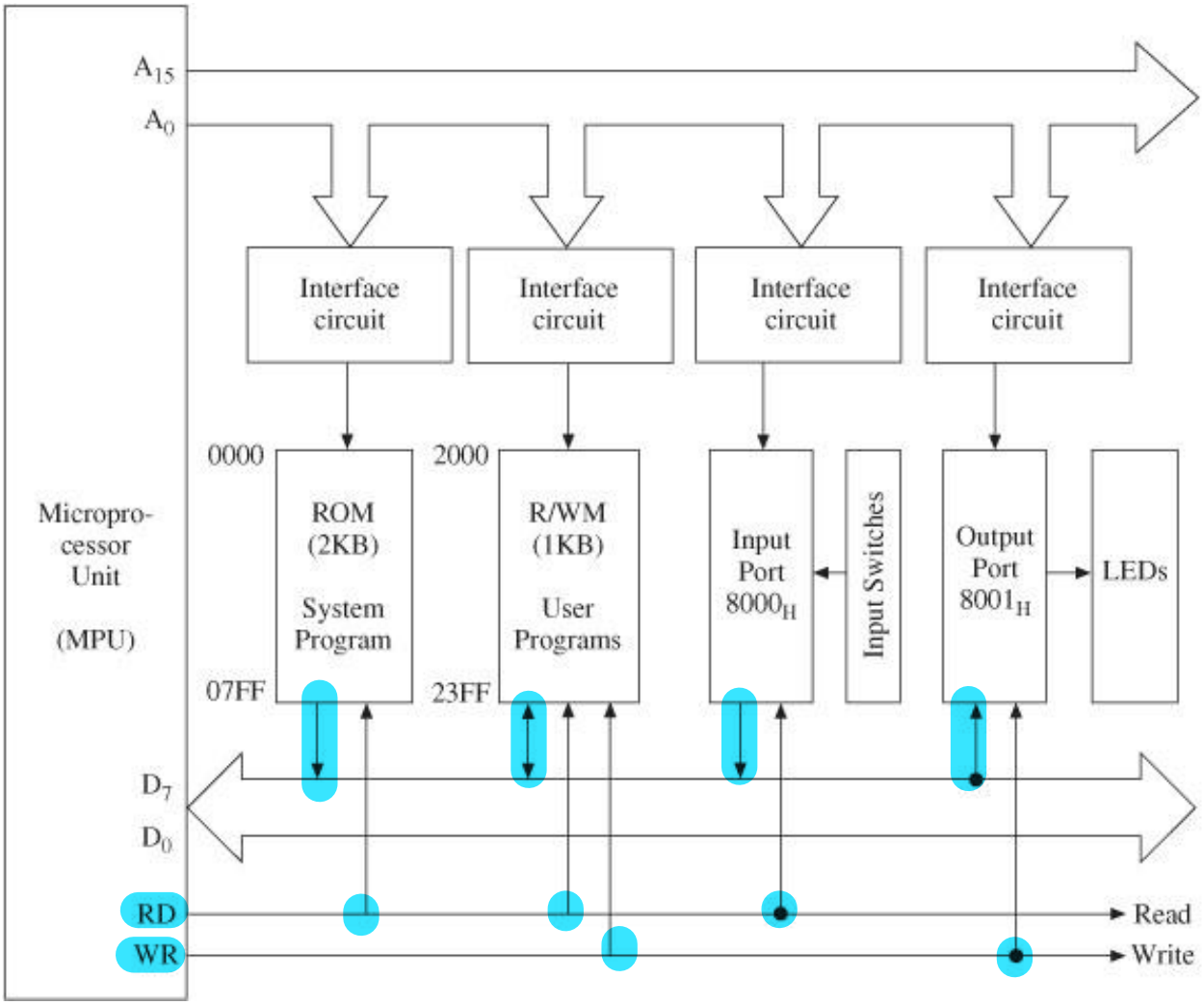
Address Bus

- Size of address bus is larger, the larger the number of devices and memory locations that can be addressed.
  - Example：8 bits（small）, 16 bits, 32 bits, 64 bits（large）.
  - A 16-bit address bus can indicate 2^{16}=64K bytes of addressable memory locations.
  - Regardless of the size of the data bus.
- Address bus is unidirectional.
- The number of address lines determines the number of locations with which a CPU can communicate.

In modern computer systems,

- CPU, memory, and a number of device controllers are connected to the system bus.
- I/O devices and the CPU can execute concurrently.
- each device controller is in charge of a particular device type.
- each device controller has a local buffer (a small number of registers).
- CPU moves data from main memory to local buffer, or from local buffer to main memory.
- data is input from the device to local buffer of controller, or output from local buffer to the device.
- device controller can inform CPU that it has finished its operation. (by using interrupt)

Example Microprocessor System



1.6 Computer operation

The operation of a computer can be summarized as follows:

- The computer accepts information in the form of programs and data through input units and stores them in the memory.
- Fetch the instructions one-by-one into the CPU.
- Decode and then execute the instructions.
- Information stored in the memory is fetched, under program control, into ALU, where it is processed.
- Processed information leaves the computer through output units.
- All activities inside the machine are directed by the control unit.

1.6.1 CPU

Execute the program by running the instructions one-by-one.

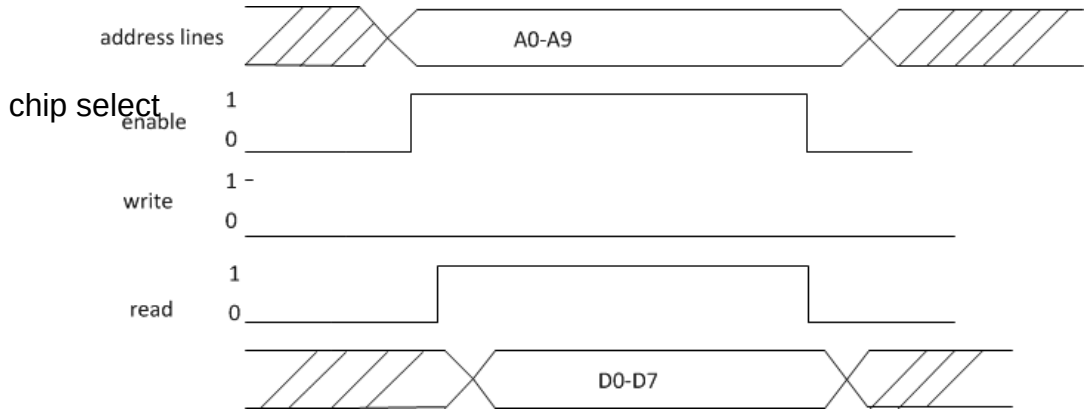
1. Load an instruction from memory to CPU.
2. Execute the instruction.
3. During the execution, the CPU may need to read or write the data from or to the memory.

Questions: How to access an instruction or data item from the memory?

How to make sure that the instructions are executed in a correct order?

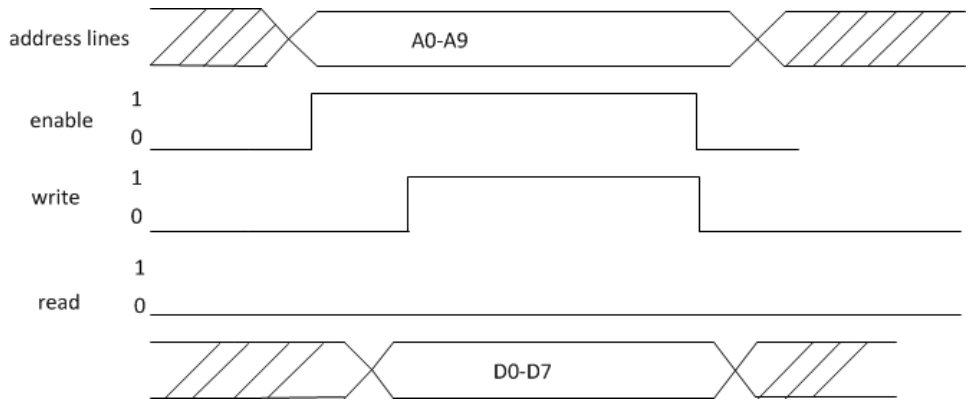
Example - CPU Read a byte from memory (read instruction or data item)

- CPU puts the address of the location on address bus
- The decoding circuit selects the chip
- CPU sets the read signal
- The memory chip then puts the content of the selected location on the data bus
- CPU now can access the data
- And then, … …

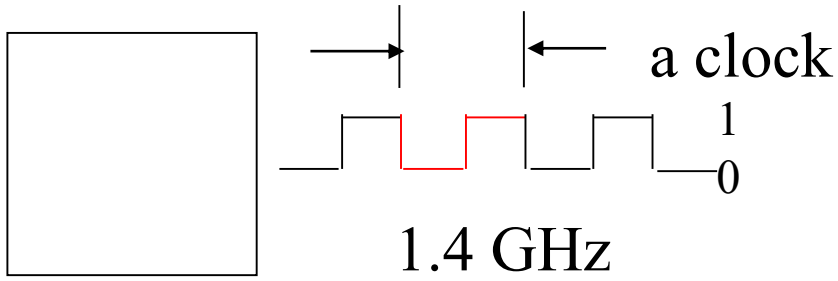


Example - CPU Write a byte to memory (write data item)

- CPU puts the address of the location on address bus
- The decoding circuit (set the enable pin) selects the chip
- CPU puts the data item on data bus
- CPU sets the write signal.
- The memory chip then takes the data from the data bus and then saves it to the selected location (indicated by the address)
- And then …, …,



- CPU performs the operations in the step-by-step manner.
- A CPU can be considered as a synchronized sequence logic circuit.
- Clock is used to synchronize work of the components on the machine.
- Clock decides the performance of the computer.



1.6.2 software

- Machine Language: binary instructions
  - Difficult to decipher and write
  - Prone to cause many errors in writing
  - All programs converted into the machine language of a processor for execution

Instruction	Hex	Mnemonic	Description	Processor
10000000	80	ADD B	Add reg B to Acc	Intel 8085
00101000	28	ADD A, R0	Add Reg R0 to Acc	Intel 8051
00011011	1B	ABA	Add Acc A and B	Motorola 6811

- Assembly Language: machine instructions represented in mnemonics
  - Has one-to-one correspondence with machine instructions
  - Efficient in execution and use of memory; machine-specific and not easy to troubleshoot

- High-Level Languages (such as BASIC, C, and C++)
  - Written in statements of spoken languages (such as English)
    - machine independent
    - easy to write and troubleshoot
    - requires large memory and less efficient in execution

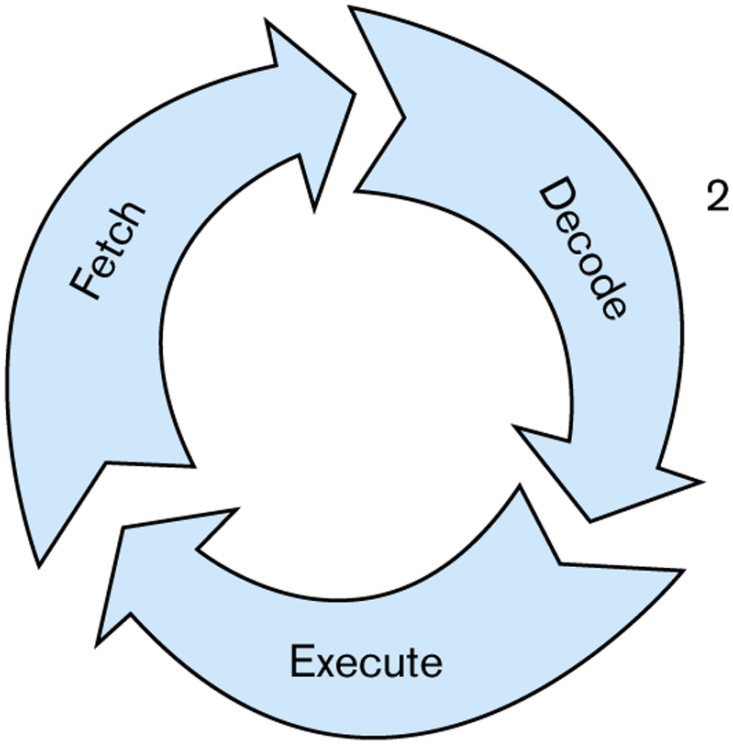
1.6.3 machine cycle

- Every instruction in memory is executed by three steps:

  Fetch → Decode → Execute

  – Each instruction has its micro-instruction (or micro-operations).
    - A micro-operation is an elementary operation that can be performed in parallel during one clock pulse period.
  – CPU has separate inside units for performing fetch/decode/execution.
  – The instruction decoder is to interpret the instruction fetched into the CPU.

1. Retrieve the next instruction from memory (as indicated by the program counter) and then increment the program counter.



2. Decode the bit pattern in the instruction register.

by instruction decoder

3. Perform the action requested by the instruction in the instruction register.

by ALU

Program Example

Action W: working register	**Machine Code**
Move value 21H into register W	0E21
Add value 42H to register W	0F42
Add value 12H to register W	0F12
No operation	0000
GOTO 0000	EF00 F000

```
Main:   ORG         0x0000
        MOVLW       0x21
        ADDLW       0x42
        ADDLW       0x12
        NOP
        GOTO        Main
        END
```

Inside Memory

Memory address (1 byte)	Contents (16bit -> 2 byte)	Meaning
0000	0E21	instruction for loading a value to W register
0002	0F42	instruction for adding a value into W register
0004	0F12	instruction for adding a value into W register
0006	0000	No operation
0008	EF00	GO Back 0000
	F000	

Actions Performed by the CPU (1/2) (each step spends a number of clock cycles)

program counter

1. The **PC** is reset to the value 0000H, indicating the address of the first instruction code to be execution.

PC=0000

2. The CPU puts 0000H on the address bus and sends it out. The **PC** is added by 2.

PC=0002

3. The memory circuitry finds 0000H while the CPU activates the READ signal, indicating to memory that CPU wants the two bytes at location 0000H.
4. The content of memory locations 0000H and 0001H, which is 0E21, is put on the data bus and brought into the CPU.

PC=0002

5. The CPU decodes the instruction 0E21

Actions Performed by the CPU (2/2)

6. The CPU knows that it is a move instruction and the data is in the second part of the instruction. The CPU performs the moving instruction.

PC=0002

W=21

7. The CPU fetches the memory location 0002H.

The CPU *fetches* instruction 0F42. The PC adds 2.

PC=0004

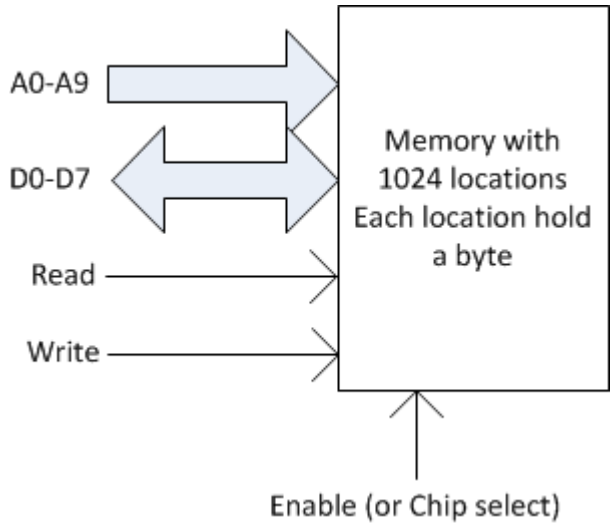
The CPU *decodes* 0F42. It is an ADD instruction. The data 42 is in the second part of the instruction. The destination is W register.

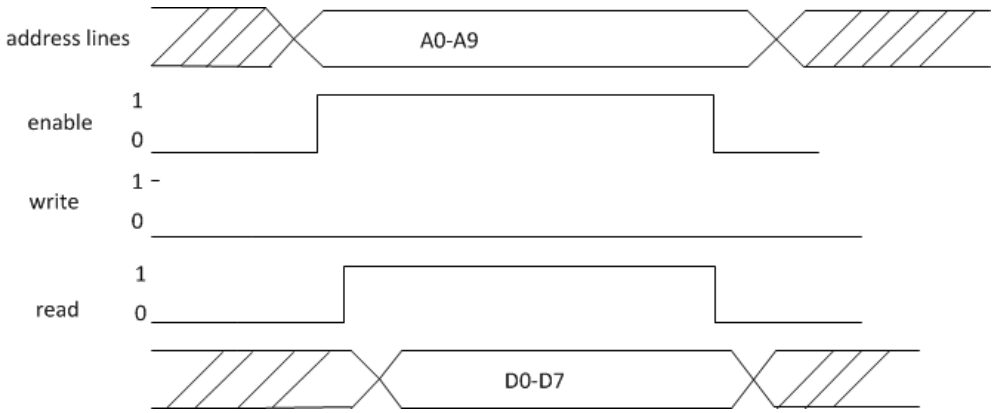
The ALU *executes* the add instruction and sets the result (63H) to register W.

PC=0004

W=63

Memory

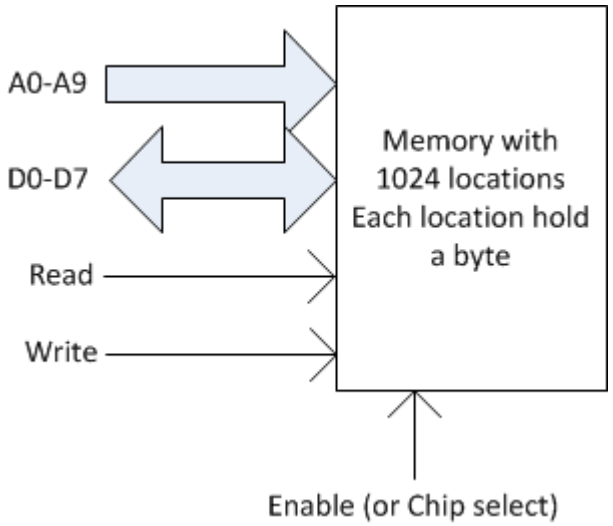


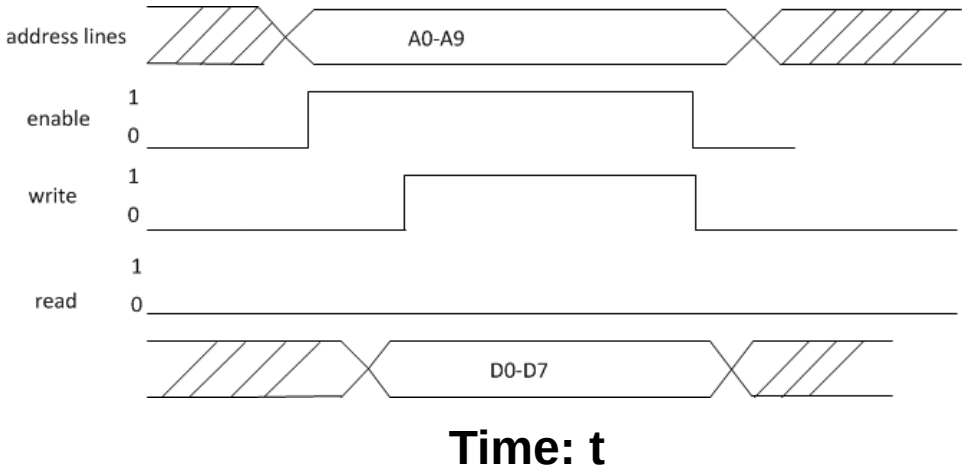


Read a byte from a location

- External circuit sets the address of the location to address lines
- External circuit enables the chip enable
- External circuit sets read signal
- The memory chip then puts the content of the selected location on D0-D7.
- Now external circuit can access the data from D0-D7

Memory





Write a byte to a location

- External circuit sets the address of the location
- External circuit enables the chip
- External circuit puts the data on D0-D7
- External circuit sets the write signal
- The memory chip then saves incoming data on the corresponding location.

The Operation of Bus

- The CPU puts the address on the address bus, and the decoding circuitry finds the device.
- The CPU uses the data bus either to get data from that device or to send data to it.
- The control bus is used to provide read or write signals.
- The address bus and data bus determine the capacity of a given CPU.

Summary

- understand the structure of computer and its major components
- learn how computer executes program
- history of computer
- review number systems and basic logic circuits