OUTLINES

- Numbering and coding systems
- Digital primer
- Inside the computer

NUMBERING AND CODING SYSTEMS

Decimal and Binary Number Systems

- Human beings use base 10 (*decimal*) arithmetic
  - There are 10 distinct symbols, 0, 1, 2, …, 9
- Computers use base 2 (*binary*) system
  - There are only 0 and 1
  - These two binary digits are commonly referred to as *bits*

NUMBERING AND CODING SYSTEMS

Converting from Decimal to Binary

- Divide the decimal number by 2 repeatedly
- Keep track of the remainders
- Continue this process until the quotient becomes zero
- Write the remainders in reverse order to obtain the binary number

Ex. Convert 25_{10} to binary

		Quotient	Remainder	
25/2	=	12	1	LSB (least significant bit)
12/2	=	6	0	
6/2	=	3	0	
3/2	=	1	1	
1/2	=	0	1	MSB (most significant bit)

Therefore $25_{10} = 11001_2$

NUMBERING AND CODING SYSTEMS

Converting from Binary to Decimal

- Know the weight of each bit in a binary number
- Add them together to get its decimal equivalent

Ex. Convert 11001_2 to decimal

Weight:	2^4	2^3	2^2	2^1	2^0
Digits:	1	1	0	0	1
Sum:	16 +	8 +	0 +	0 +	1 = 25_{10}

- Use the concept of weight to convert a decimal number to a binary directly

Ex. Convert 39_{10} to binary

32 + 0 + 0 + 4 + 2 + 1 = 39

Therefore, $39_{10} = 100111_2$

NUMBERING AND CODING SYSTEMS

Hexadecimal System

- Base 16, the *hexadecimal* system, is used as a convenient representation of binary numbers
  - ex.

    It is much easier to represent a string of 0s and 1s such as 100010010110 as its hexadecimal equivalent of 896H

Decimal	Binary	Hex
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

NUMBERING AND CODING SYSTEMS

Converting between Binary and Hex

- To represent a binary number as its equivalent hexadecimal number
  - Start from the right and group 4 bits at a time, replacing each 4-bit binary number with its hex equivalent

Ex. Represent binary 100111110101 in hex

	1001	1111	0101
=	9	F	5

- To convert from hex to binary
  - Each hex digit is replaced with its 4-bit binary equivalent

Ex. Convert hex 29B to binary

	2	9	B
=	0010	1001	1011

NUMBERING AND CODING SYSTEMS

Converting from Decimal to Hex

- Convert to binary first and then convert to hex
- Convert directly from decimal to hex by repeated division, keeping track of the remainders

Ex. Convert 45_{10} to hex

32	16	8	4	2	1	
1	0	1	1	0	1	32 + 8 + 4 + 1 = 45

$45_{10} = 0010\ 1101_2 = 2D_{16}$

Ex. Convert 629_{10} to hex

512	256	128	64	32	16	8	4	2	1
1	0	0	1	1	1	0	1	0	1

$629_{10} = 512+64+32+16+4+1 = 0010\ 0111\ 0101_2 = 275_{16}$

NUMBERING AND CODING SYSTEMS

Converting from Hex to Decimal

- Convert from hex to binary and then to decimal
- Convert directly from hex to decimal by summing the weight of all digits

Ex. $6B2_{16} = 0110\ 1011\ 0010_2$

1024	512	256	128	64	32	16	8	4	2	1
1	1	0	1	0	1	1	0	0	1	0

$1024 + 512 + 128 + 32 + 16 + 2 = 1714_{10}$

NUMBERING AND CODING SYSTEMS

Addition of Hex Numbers

- Adding the digits together from the least significant digits
  - If the result is less than 16, write that digit as the sum for that position
  - If it is greater than 16, subtract 16 from it to get the digit and carry 1 to the next digit

Ex. Perform hex addition: 23D9 + 94BE

23D9	LSD:	9 + 14 = 23	23 – 16 = 7 w/ carry
+ 94BE		1 + 13 + 11 = 25	25 – 16 = 9 w/ carry
B897		1 + 3 + 4 = 8	
	MSD:	2 + 9 = B	

NUMBERING AND CODING SYSTEMS

Subtraction of Hex Numbers

- If the second digit is greater than the first, borrow 16 from the preceding digit

Ex. Perform hex subtraction: 59F – 2B8

59F	LSD:	15 – 8 = 7
– 2B8		9 + 16 – 11 = 14 = E_{16}
2E7		5 – 1 – 2 = 2

NUMBERING AND CODING SYSTEMS

ASCII Code

- The ASCII (pronounced “ask-E”) code assigns binary patterns for
  - Numbers 0 to 9
  - All the letters of English alphabet, uppercase and lowercase
  - Many control codes and punctuation marks
- The ASCII system uses 7 bits to represent each code

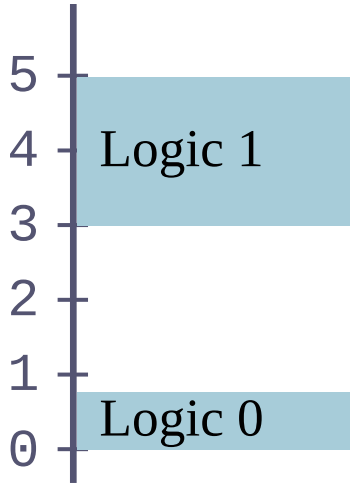
Selected ASCII codes

Hex	*Symbol*	*Hex*	*Symbol*
41	A	61	a
42	B	62	b
43	C	63	c
44	D	64	d
…	…	…	…
59	Y	79	y
5A	Z	7A	z

DIGITAL PRIMER

Binary Logic

- Two voltage levels can be represented as the two digits 0 and 1
- Signals in digital electronics have two distinct voltage levels with built-in tolerances for variations in the voltage
- A valid digital signal should be within either of the two shaded areas



DIGITAL PRIMER

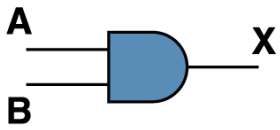
Logic Gates

- AND gate

Boolean Expression

$X = A \cdot B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

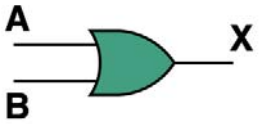
Computer Science Illuminated, Dale and Lewis

- OR gate

Boolean Expression

$X = A + B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Computer Science Illuminated, Dale and Lewis

DIGITAL PRIMER

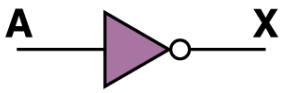
Logic Gates (cont')

- Tri-state buffer
- Inverter

Boolean Expression

$X = A'$

Logic Diagram Symbol



Truth Table

A	X
0	1
1	0

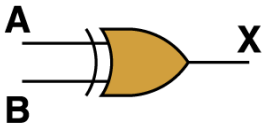
Computer Science Illuminated, Dale and Lewis

- XOR gate

Boolean Expression

$X = A \oplus B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Computer Science Illuminated, Dale and Lewis

DIGITAL PRIMER

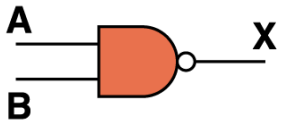
Logic Gates (cont')

NAND gate

Boolean Expression

$X = (A \cdot B)'$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

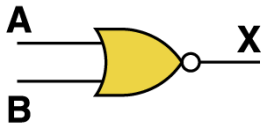
Computer Science Illuminated, Dale and Lewis

NOR gate

Boolean Expression

$X = (A + B)'$

Logic Diagram Symbol



Truth Table

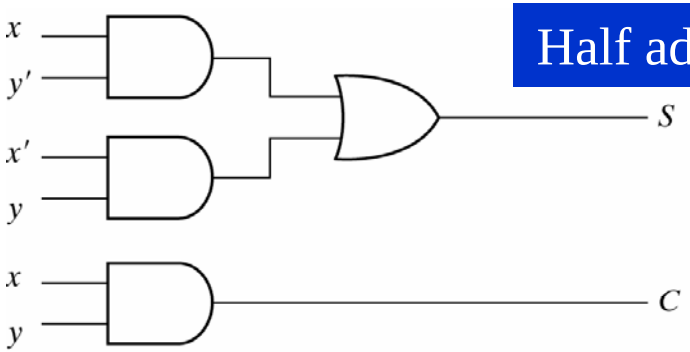
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Computer Science Illuminated, Dale and Lewis

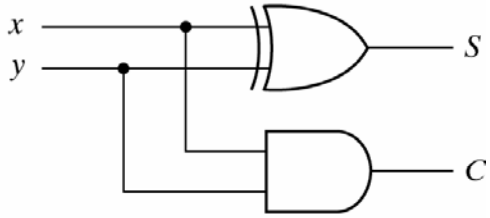
DIGITAL PRIMER

Logic Design Using Gates

Half adder

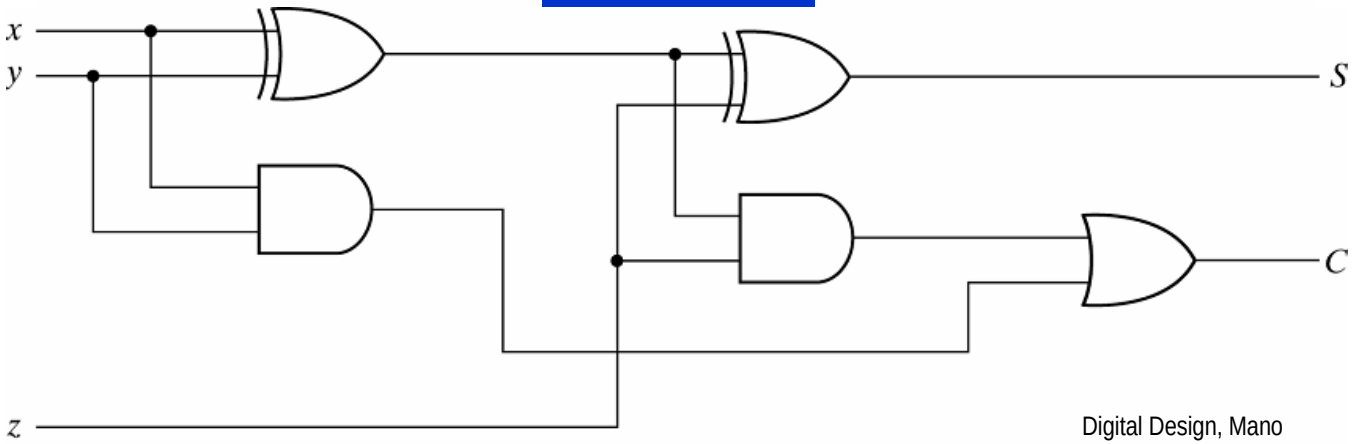


(a) $S = xy' + x'y$
$C = xy$



(b) $S = x \oplus y$
$C = xy$

Full adder

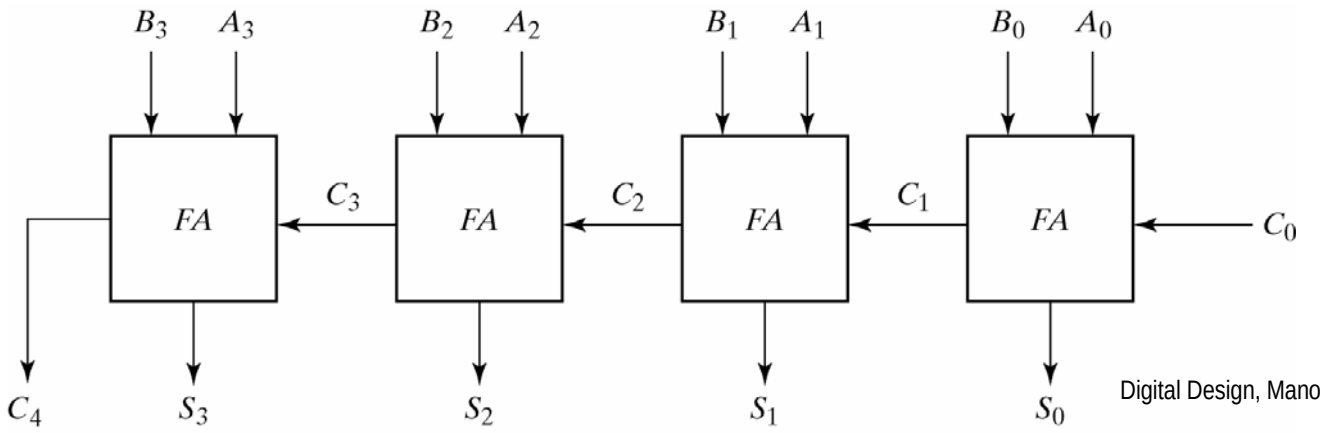


Digital Design, Mano

DIGITAL PRIMER

Logic Design Using Gates (cont')

4-bit adder



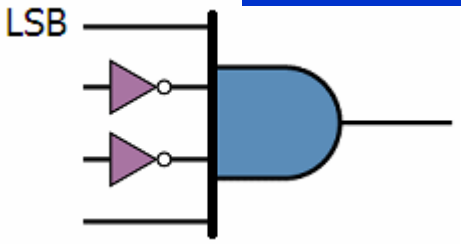
Digital Design, Mano

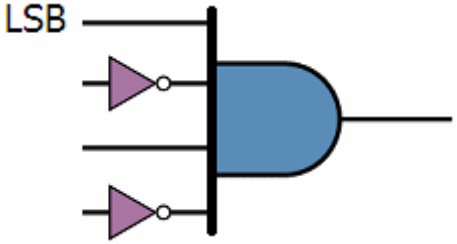
DIGITAL PRIMER

Logic Design Using Gates (cont')

- Decoders
  - Decoders are widely used for address decoding in computer design

Address Decoders





Address decoder for 9 (1001_2)

The output will be 1 if and only if the input is 1001_2

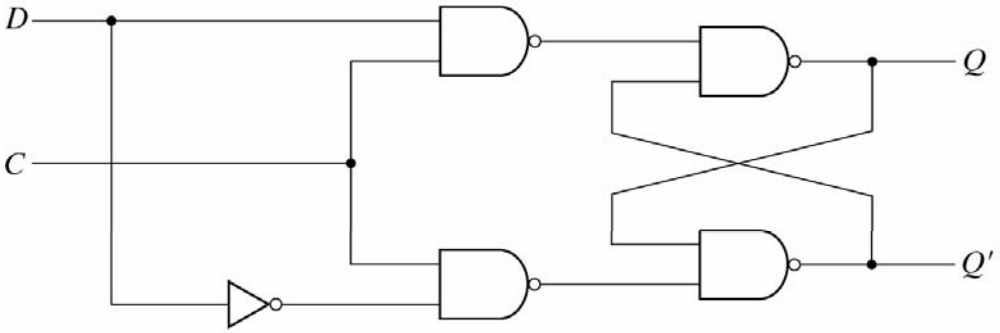
Address decoder for 5 (0101_2)

The output will be 1 if and only if the input is 0101_2

DIGITAL PRIMER

Logic Design Using Gates (cont')

- Flip-flops
  - Flip-flops are frequently used to store data

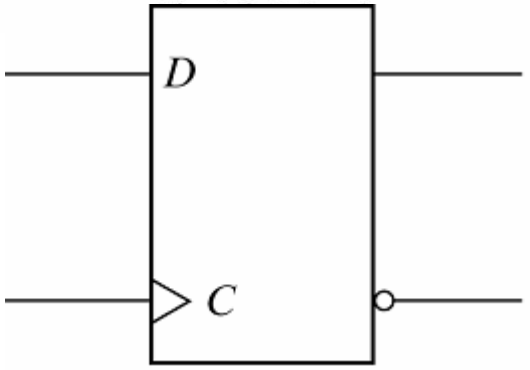


Digital Design, Mano

(a) Logic diagram

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

(b) Function table



INSIDE THE COMPUTER

Important Terminology

- The unit of data size
  - *Bit* : a binary digit that can have the value 0 or 1
  - *Byte* : 8 bits
  - *Nibble* : half of a bye, or 4 bits
  - *Word* : two bytes, or 16 bits
- The terms used to describe amounts of memory in IBM PCs and compatibles
  - *Kilobyte* (K): 2^{10} bytes
  - *Megabyte* (M) : 2^{20} bytes, over 1 million
  - *Gigabyte* (G) : 2^{30} bytes, over 1 billion
  - *Terabyte* (T) : 2^{40} bytes, over 1 trillion

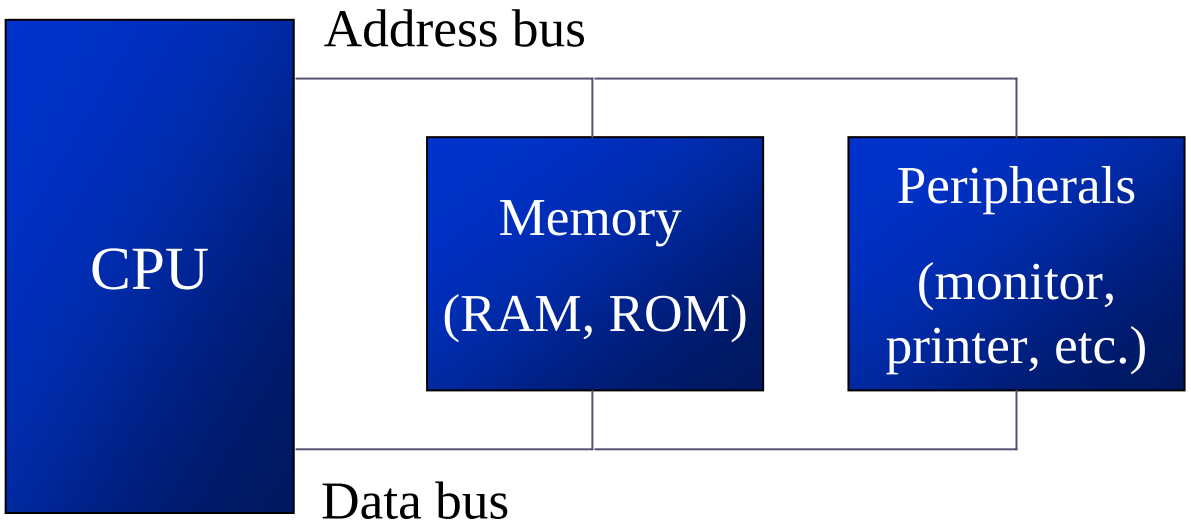
INSIDE THE COMPUTER

Internal Organization of Computers

- CPU (Central Processing Unit)
  - Execute information stored in memory
- I/O (Input/output) devices
  - Provide a means of communicating with CPU
- Memory
  - RAM (Random Access Memory) – temporary storage of programs that computer is running
    - The data is lost when computer is off
  - ROM (Read Only Memory) – contains programs and information essential to operation of the computer
    - The information cannot be changed by use, and is not lost when power is off
      - It is called *nonvolatile memory*

INSIDE THE COMPUTER

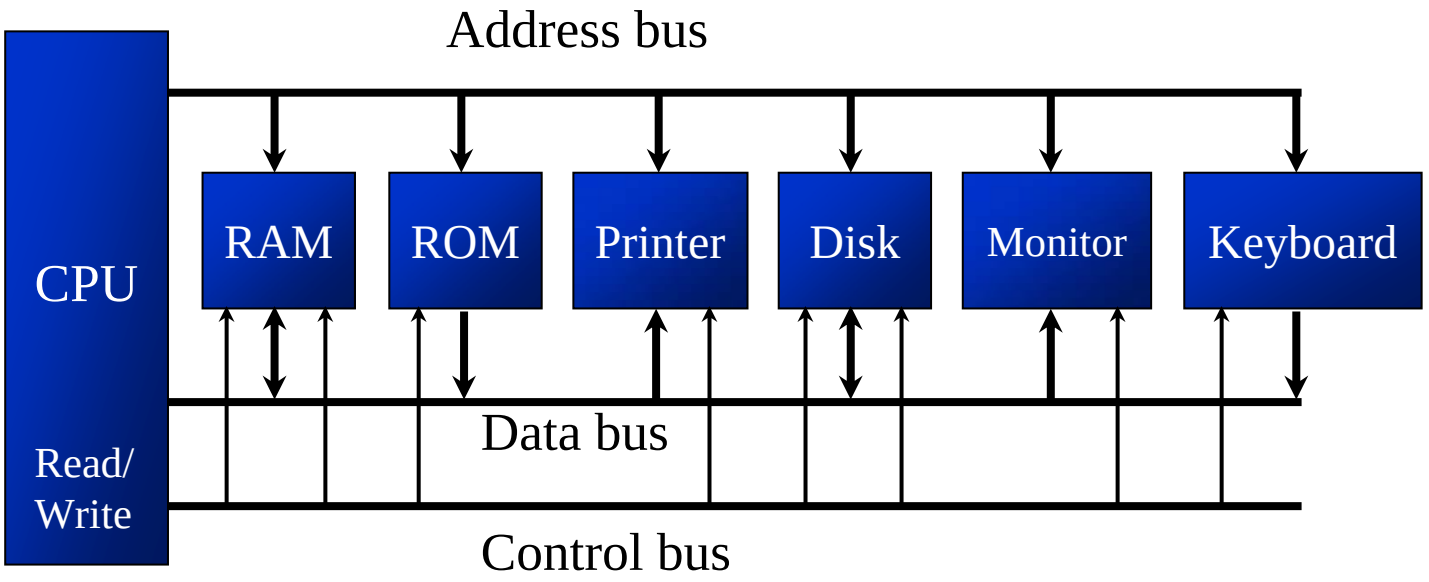
Internal Organization of Computers (cont')



INSIDE THE COMPUTER

Internal Organization of Computers (cont')

- The CPU is connected to memory and I/O through strips of wire called a *bus*
  - Carries information from place to place
    - Address bus
    - Data bus
    - Control bus



INSIDE THE COMPUTER

Internal Organization of Computers (cont')

- Address bus
  - For a device (memory or I/O) to be recognized by the CPU, it must be assigned an address
    - The address assigned to a given device must be unique
    - The CPU puts the address on the address bus, and the decoding circuitry finds the device
- Data bus
  - The CPU either gets data from the device or sends data to it
- Control bus
  - Provides read or write signals to the device to indicate if the CPU is asking for information or sending it information

INSIDE THE COMPUTER

More about Data Bus

- The more data buses available, the better the CPU
  - Think of data buses as highway lanes
- More data buses mean a more expensive CPU and computer
  - The average size of data buses in CPUs varies between 8 and 64
- Data buses are bidirectional
  - To receive or send data
- The processing power of a computer is related to the size of its buses

INSIDE THE COMPUTER

More about Address Bus

- The more address buses available, the larger the number of devices that can be addressed
- The number of locations with which a CPU can communicate is always equal to 2^x, where x is the address lines, regardless of the size of the data bus
  - ex. a CPU with 24 address lines and 16 data lines can provide a total of 2^{24} or 16M bytes of addressable memory
  - Each location can have a maximum of 1 byte of data, since all general-purpose CPUs are *byte addressable*
- The address bus is unidirectional

INSIDE THE COMPUTER

CPU's Relation to RAM and ROM

- For the CPU to process information, the data must be stored in RAM or ROM, which are referred to as *primary memory*
- ROM provides information that is fixed and permanent
  - Tables or initialization program
- RAM stores information that is not permanent and can change with time
  - Various versions of OS and application packages
  - CPU gets information to be processed
    - first form RAM (or ROM)
    - if it is not there, then seeks it from a mass storage device, called *secondary memory*, and transfers the information to RAM

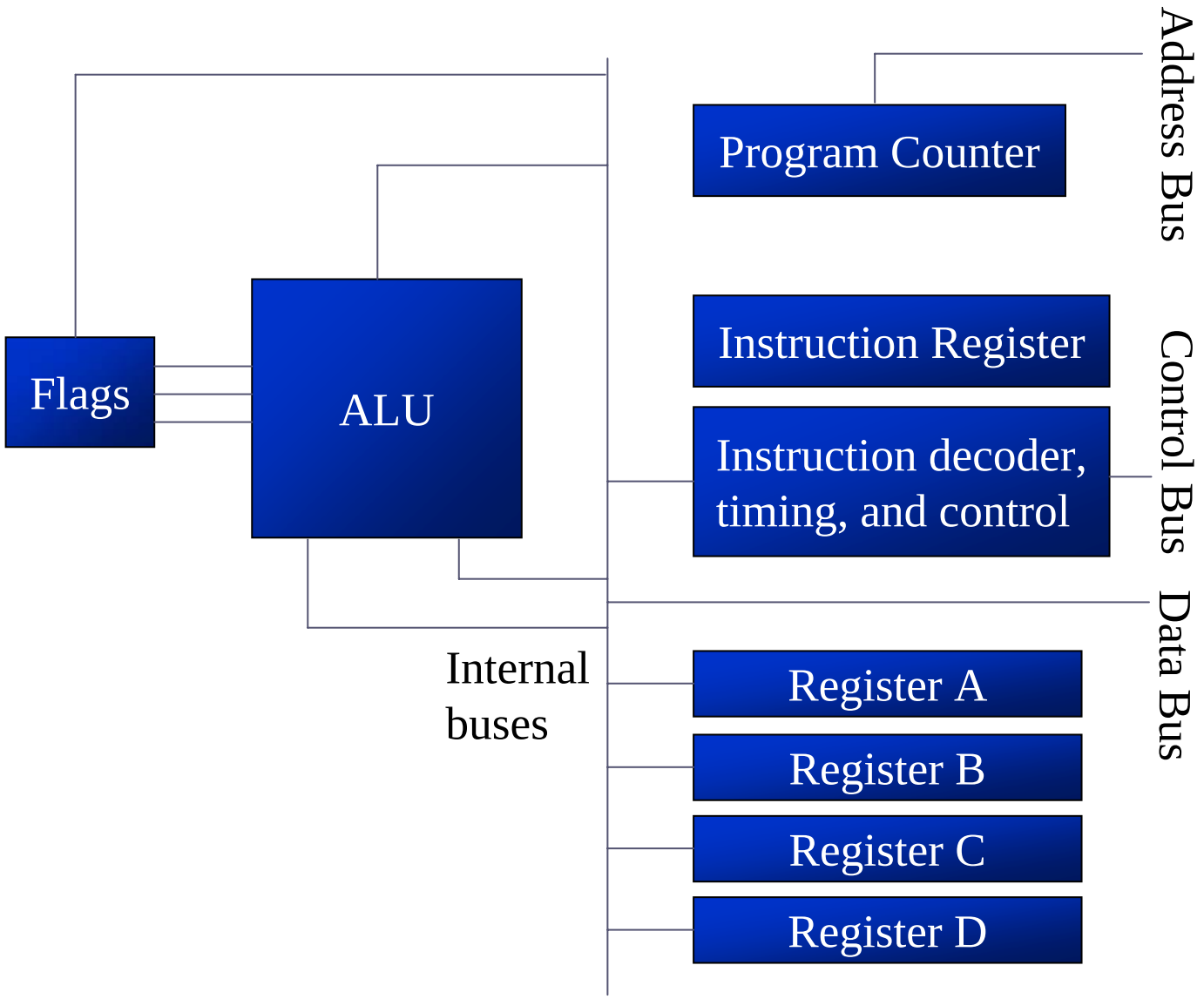
INSIDE THE COMPUTER

Inside CPUs

- Registers
  - The CPU uses registers to store information temporarily
    - Values to be processed
    - Address of value to be fetched from memory
  - In general, the more and bigger the registers, the better the CPU
    - Registers can be 8-, 16-, 32-, or 64-bit
    - The disadvantage of more and bigger registers is the increased cost of such a CPU

INSIDE THE COMPUTER

Inside CPUs (cont')



INSIDE THE COMPUTER

Inside CPUs (cont')

- ALU (arithmetic/logic unit)
  - Performs arithmetic functions such as add, subtract, multiply, and divide, and logic functions such as AND, OR, and NOT
- Program counter
  - Points to the address of the next instruction to be executed
    - As each instruction is executed, the program counter is incremented to point to the address of the next instruction to be executed
- Instruction decoder
  - Interprets the instruction fetched into the CPU
    - A CPU capable of understanding more instructions requires more transistors to design

INSIDE THE COMPUTER

Internal Working of Computers

Ex. A CPU has registers A, B, C, and D and it has an 8-bit data bus and a 16-bit address bus. The CPU can access memory from addresses 0000 to FFFFH

Assume that the code for the CPU to move a value to register A is B0H and the code for adding a value to register A is 04H

The action to be performed by the CPU is to put 21H into register A, and then add to register A values 42H and 12H

…

INSIDE THE COMPUTER

Internal Working of Computers (cont')

Ex. (cont')

Action	*Code*	*Data*
Move value 21H into reg. A	B0H	21H
Add value 42H to reg. A	04H	42H
Add value 12H to reg. A	04H	12H

Mem. addr.	*Contents of memory address*
1400	(B0) code for moving a value to register A
1401	(21) value to be moved
1402	(04) code for adding a value to register A
1403	(42) value to be added
1404	(04) code for adding a value to register A
1405	(12) value to be added
1406	(F4) code for halt

...

INSIDE THE COMPUTER

Internal Working of Computers (cont')

Ex. (cont')

The actions performed by CPU are as follows:

1. The program counter is set to the value 1400H, indicating the address of the first instruction code to be executed
2.
   - The CPU puts 1400H on address bus and sends it out
     - The memory circuitry finds the location
   - The CPU activates the READ signal, indicating to memory that it wants the byte at location 1400H
     - This causes the contents of memory location 1400H, which is B0, to be put on the data bus and brought into the CPU

...

INSIDE THE COMPUTER

Internal Working of Computers (cont')

Ex. (cont')

3.

- The CPU decodes the instruction B0
- The CPU commands its controller circuitry to bring into register A of the CPU the byte in the next memory location
  - The value 21H goes into register A
- The program counter points to the address of the next instruction to be executed, which is 1402H
  - Address 1402 is sent out on the address bus to fetch the next instruction

…

INSIDE THE COMPUTER

Internal Working of Computers (cont')

Ex. (cont')

4.

- From memory location 1402H it fetches code 04H
- After decoding, the CPU knows that it must add to the contents of register A the byte sitting at the next address (1403)
- After the CPU brings the value (42H), it provides the contents of register A along with this value to the ALU to perform the addition
  - It then takes the result of the addition from the ALU's output and puts it in register A
  - The program counter becomes 1404, the address of the next instruction

…

INSIDE THE COMPUTER

Internal Working of Computers (cont')

Ex. (cont')

5.

- Address 1404H is put on the address bus and the code is fetched into the CPU, decoded, and executed
  - This code is again adding a value to register A
  - The program counter is updated to 1406H

6.

- The contents of address 1406 are fetched in and executed
- This HALT instruction tells the CPU to stop incrementing the program counter and asking for the next instruction