Module 1

Introduction to Microcontrollers and Microprocessors

Binary and Hexadecimal Systems

- Conversion to decimal:
  - 110.101 b = ?
  - 6A.C h = ?
- Conversion from decimal
  - for a whole number: divide by the radix and save the remainder as the significant digits
  - 10 = ? b
  - 10 = ? $_8$
- Converting from a decimal fraction
  - multiply the decimal fraction by the radix
  - save the whole number part of the result
  - repeat above until fractional part of step 2 is 0
  - 0.125 = ? b
  - 0.78125 = ? h

- 110.101 b = 6.625
- 6A.C = 106.75

- 10 = 1010 b

- 10 = 12 $_8$

- 0.125 = 0.001 b
- 0.78125 = 0.C8 h

Base 16 Number systems

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

- Represent 100111110101b in hex

  Group them

  1001 1111 0101

  9 F 5 hex

- Convert 1714d to binary

  1714 = 16*107 +2

  107 = 16*6 + 11

  1714 = 6B2 h

  = 0110 1011 0010 b

Signed Representation-Two's Complement

- If the number is positive make no changes
- If the number is negative, complement all bits and add by 1
  - -6 => $\overline{0000\ 0110}$ + 1 = 1111 1001 + 1 = FAh
- 8 bit signed numbers
  - 0 to 7Fh (+127) are positive numbers
  - 80h (-128) to FFh (-1) are negative numbers
- Conversion of signed binary numbers to their decimal equivalent
  - 1101 0001
    - $\overline{1101\ 0001}$ + 1 = 0010 1110 + 1 = 0010 1111 = 2Fh => -47
  - 1000 1111 0101 1101 (16 bit signed number)
    - 0111 0000 1010 0010 + 1 = 0111 0000 1010 0011 = 70A3h => -28835
- Two's complement arithmetic
  - +14 - 20
  - 0000 1110 + $\overline{0001\ 0100}$ + 1 = FAh
- *Overflow:* Whenever two signed numbers are added or subtracted the possibility exists that the result may be too large for the number of bits allocated Ex: +64 +96 using 8-bit signed numbers

Two's Complement

Decimal	Binary	Hex
-128	1000 0000 b	80h
-127	1000 0001b	81h
-126	1000 0010b	82h
…		…
-2	1111 1110b	FEh
-1	1111 1111b	FFh
0	0000 0000b	00h
1	0000 0001b	01h
…		…
+127	0111 1111b	7Fh

Numbers in the range from -2^7 to 2^7-1
are represented by 8 bit signed arithmetic

ASCII

- The standard for text
- In this code each letter of the alphabet, punctuation mark, and decimal number is assigned a unique 7-bit code number
- With 7 bits, 128 unique symbols can be coded
- Often a zero is placed in the most significant bit position to make it an 8-bit code
  - e.g., Uppercase A 41h
- Digits 0 through 9 are represented by ASCII codes 30h to 39h

ASCII TABLE

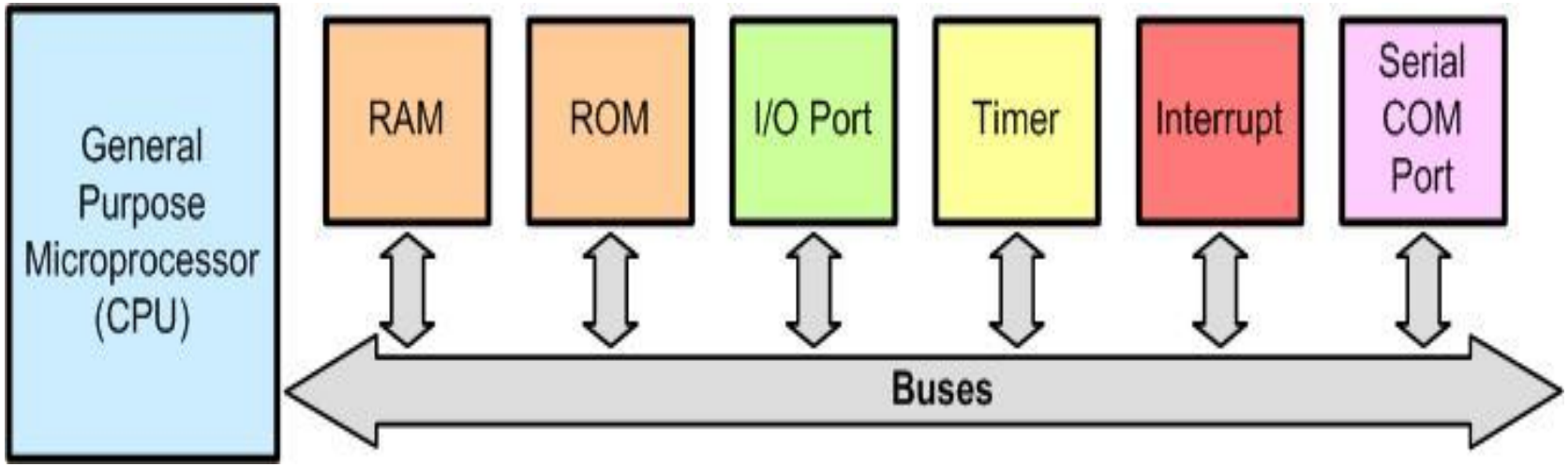
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

BCD

- BCD code provides a way for decimal numbers to be encoded in binary form that is easily converted back to decimal
  - 26 (unsigned binary)=> 1Ah = 0001 1010 b
  - 26 (BCD)=>26h=0010 0110 b
  - 243 (unsigned binary)=> F3h= 1111 0011 b
  - 243 (BCD)=>243h=> 0010 0100 0011 b
- Unpacked BCD: One byte to store each digit
  - 243 (Unpacked BCD)=> 02 04 03h
    =>00000010 00000100 00000011 b

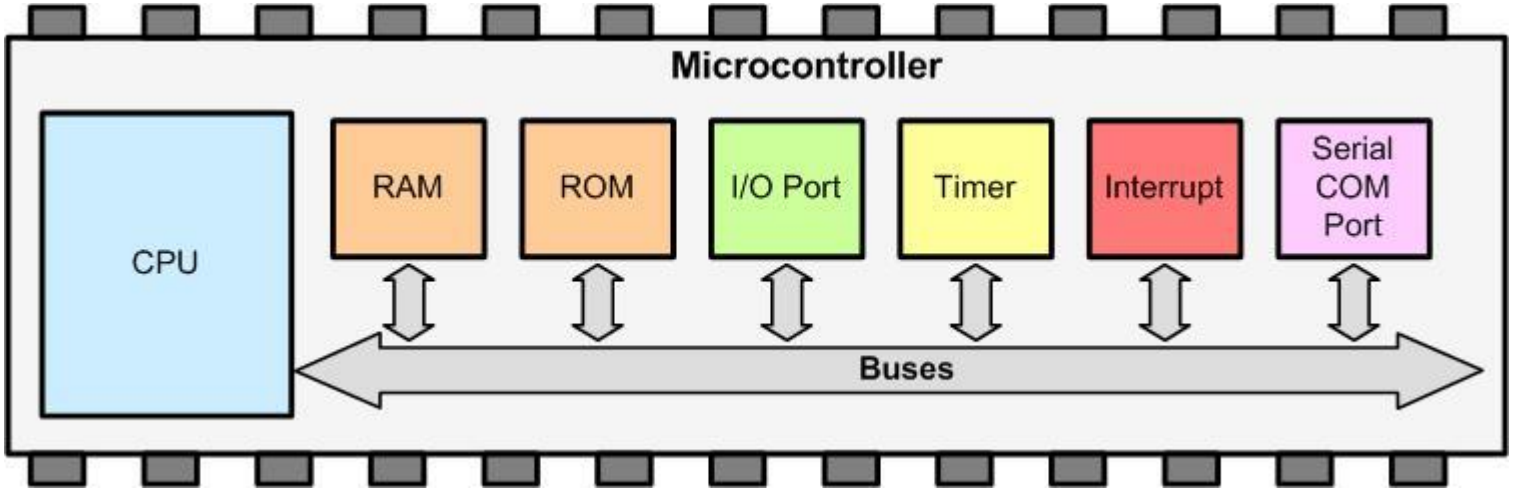
General Purpose Microprocessors

Microprocessors lead to versatile products



- These general microprocessors contain no RAM, ROM, or I/O ports on the chip itself
- Intel's x86 family (8088, 8086, 80386, 80386, 80486, Pentium
- ARM Cortex A family

Microcontrollers



- A microcontroller has a CPU in addition to a fixed amount of RAM, ROM, I/O ports on one single chip; this makes them ideal for applications in which cost and space are critical
- **Intel's 8051**, PIC 16X, ATMEL Atmega, **ARM Cortex-M**

Microprocessor vs microcontroller

Microprocessor

- CPU is stand-alone, RAM, ROM, I/O, timer are separate
- Designer can decide on the amount of ROM, RAM and I/O ports.
- General-purpose
- Expensive
- Higher processing power

Microcontroller

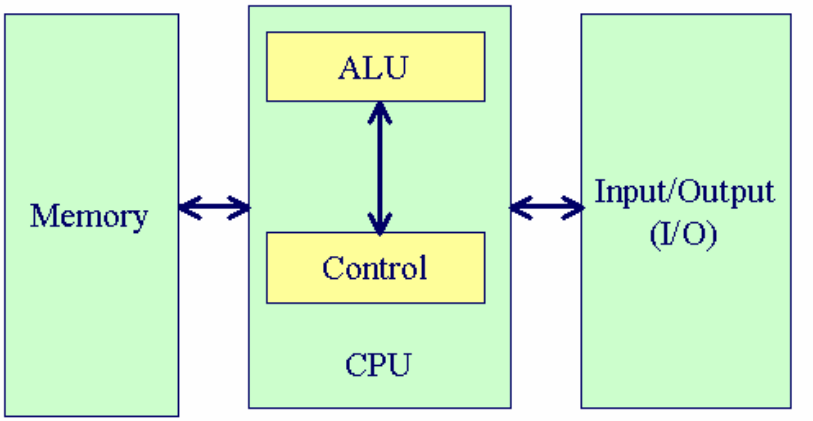
- CPU, RAM, ROM, I/O and timer are all on a single chip
- Fix amount of on-chip ROM, RAM, I/O ports
- Single-purpose
- Inexpensive
- For applications in which cost, power and space are critical

Internal Organization of Computers

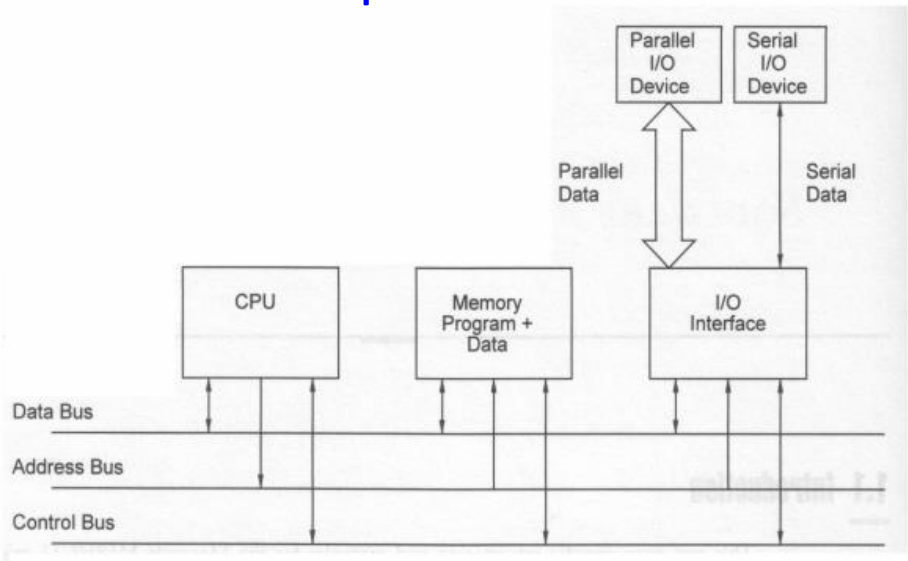
- Different parts of a computer
  - I/O
    - Input e.g. Keyboard, Mouse, Sensor
    - Output e.g. LCD, printer, hands of a robot
  - Memory
  - CPU
- Connecting the different parts
  - Connecting memory to CPU
  - Connecting I/O to CPU
- Need to understand ***how computers work*** in order to understand ***how microprocessor or microcontroller based systems work***

Common Computer Organization

Computer Organization



Main Computer Buses



ALU (Arithmetic Logic Unit) executes arithmetic and logic operations (for example ADD, SHIFT, AND, OR, etc) on certain on-chip registers.

CPU (Central Processing Unit) is the combination of the control logic, associated registers and the arithmetic logic unit (brains of the computer).

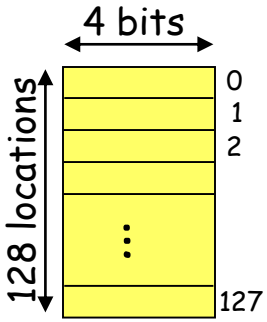
Address bus: carries the address of a unique memory or input/output (I/O) device

Data bus: carries data stored in memory (or an I/O device) to the CPU or from the CPU to the memory (or I/O device)

Control bus: is a collection of control signals that coordinate and synchronize the whole system

Memory

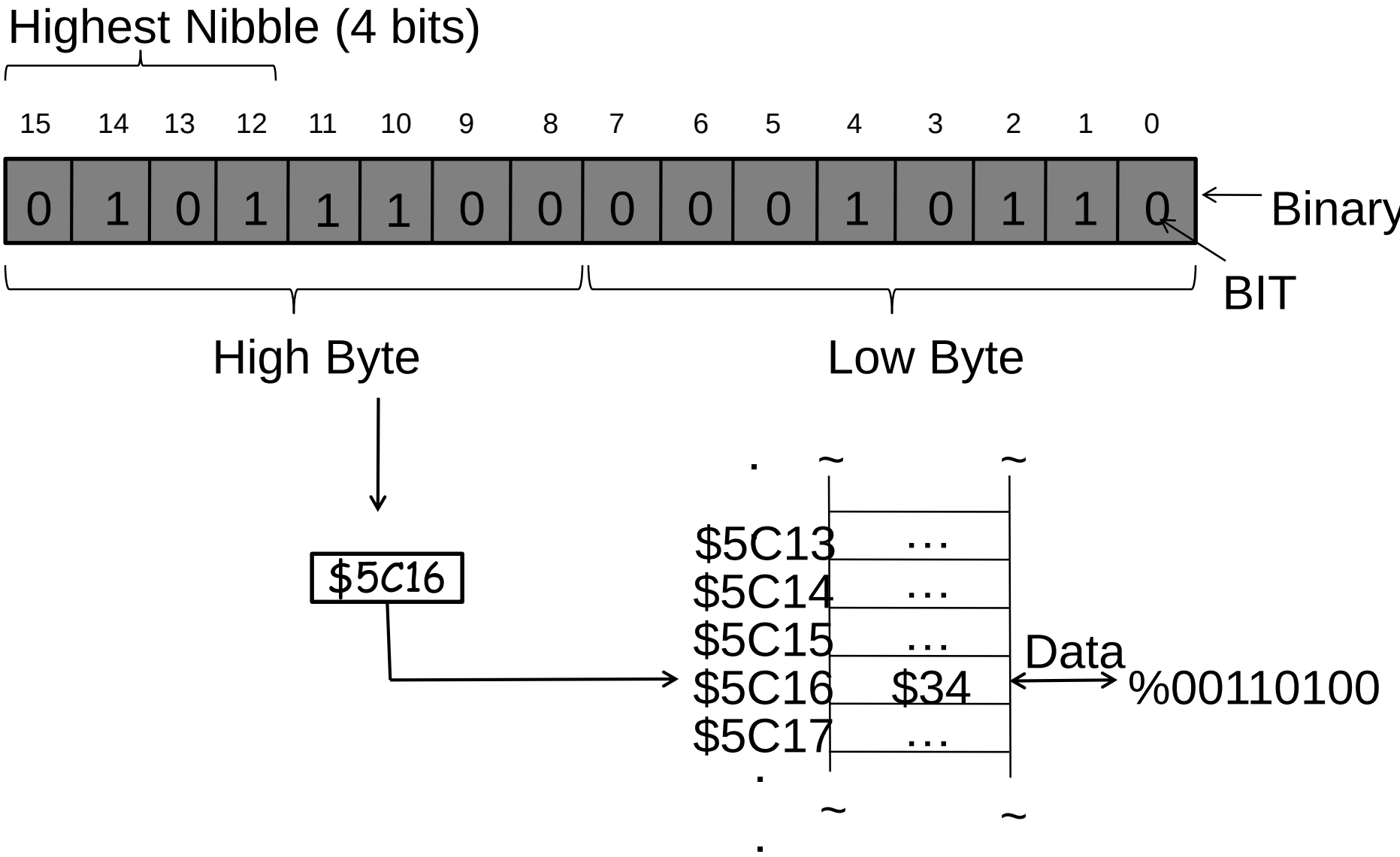
- Capacity
  - The number of bits that a memory can store.
    - e.g. 128 Kbits, 256 Mbits
- Organization
  - How the locations are organized
    - e.g. a 128 x 4 memory has 128 locations, 4 bits each
- Access time (a performance metric)
  - How long it takes to get data from memory



Semiconductor Memories

- ROM
  - Mask ROM
  - PROM (Programmable ROM)
  - EPROM (Erasable PROM)
  - EEPROM (Electronic Erasable PROM)
  - Flash EPROM
- RAM
  - SRAM (Static RAM)
  - DRAM (Dynamic RAM)
  - NV-RAM (Nonvolatile RAM)

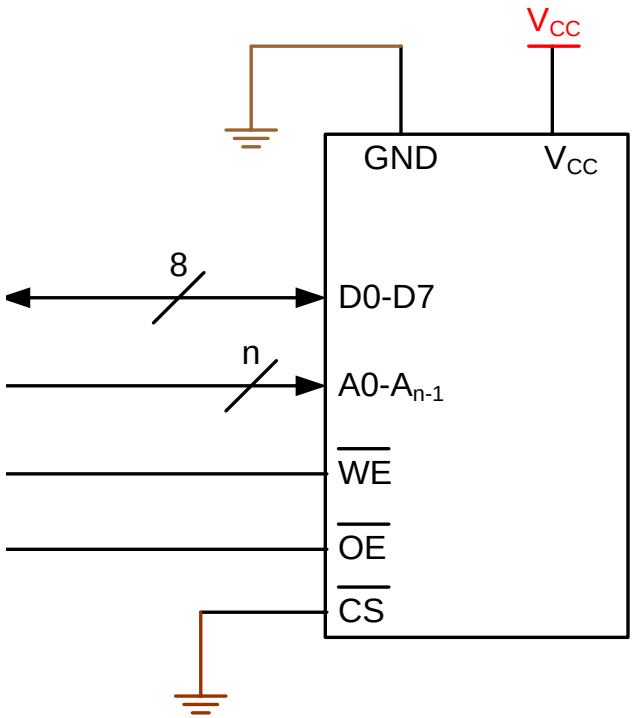
Memory Access



CPU

- Tasks:
  - It should execute instructions
    - It should fetch (recall) the instructions one after another and execute them

Connecting Memory to CPU

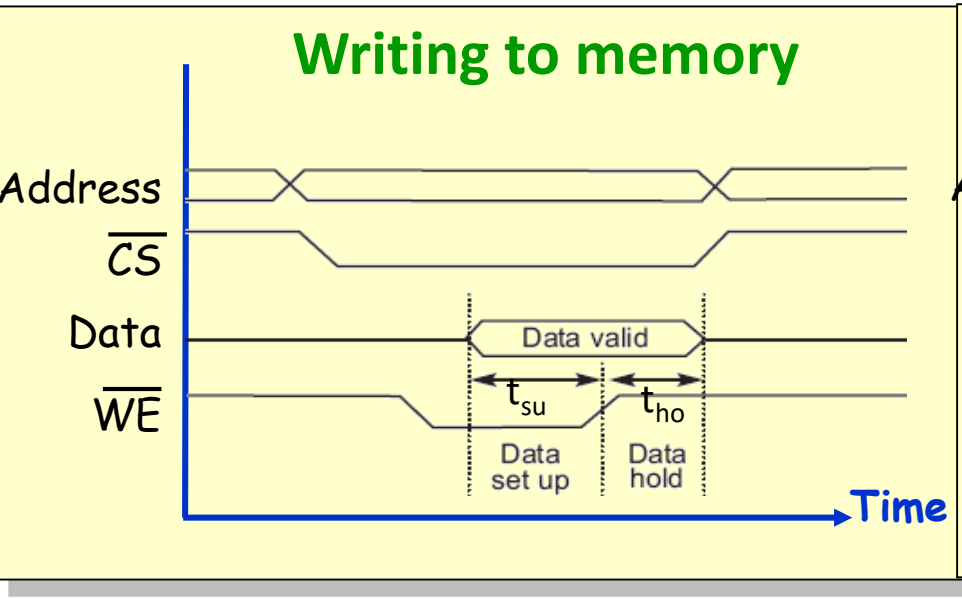


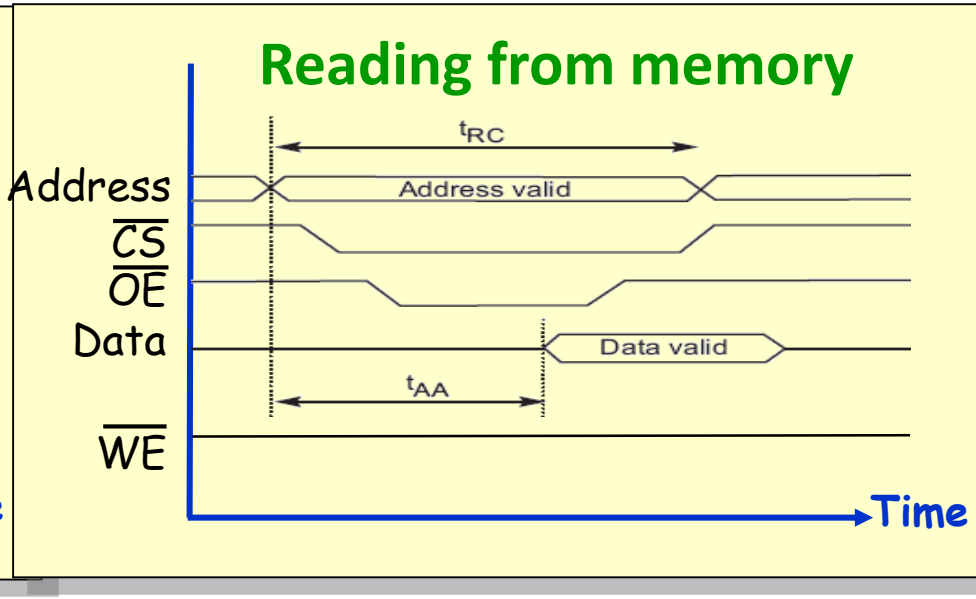
$\overline{CS}$: Active low Chip Select

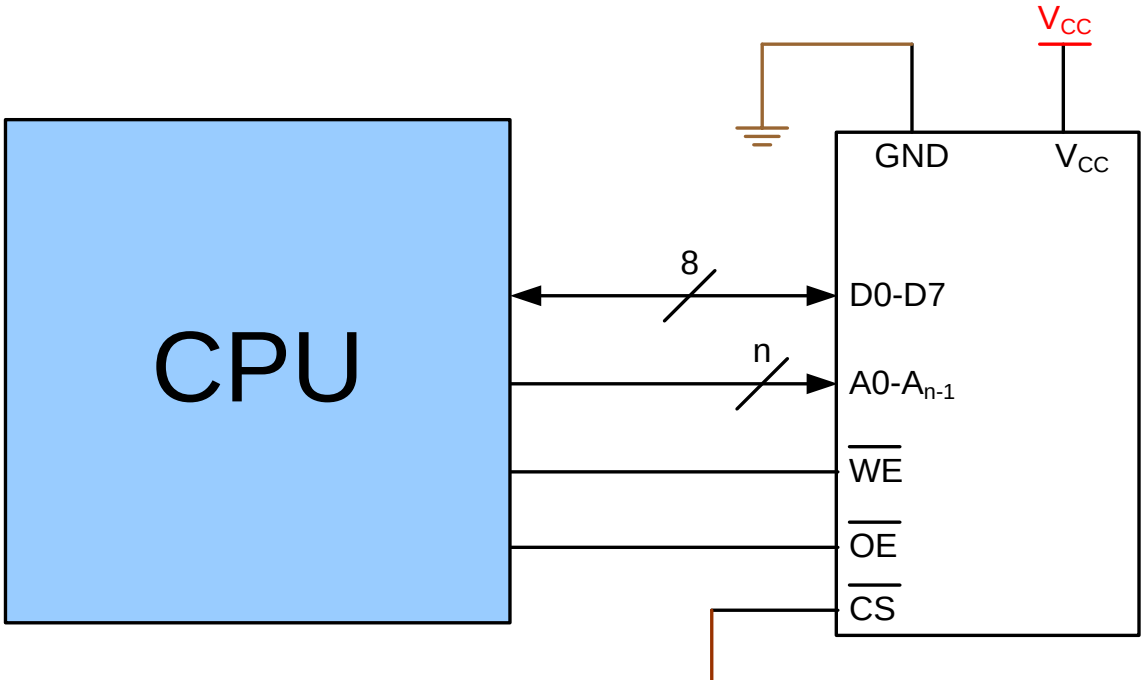
$\overline{WE}$: Active low Write Enable

$\overline{OE}$: Active low Output Enable

Connecting Memory to CPU







t_{su}: Min data setup time before WE's rising edge

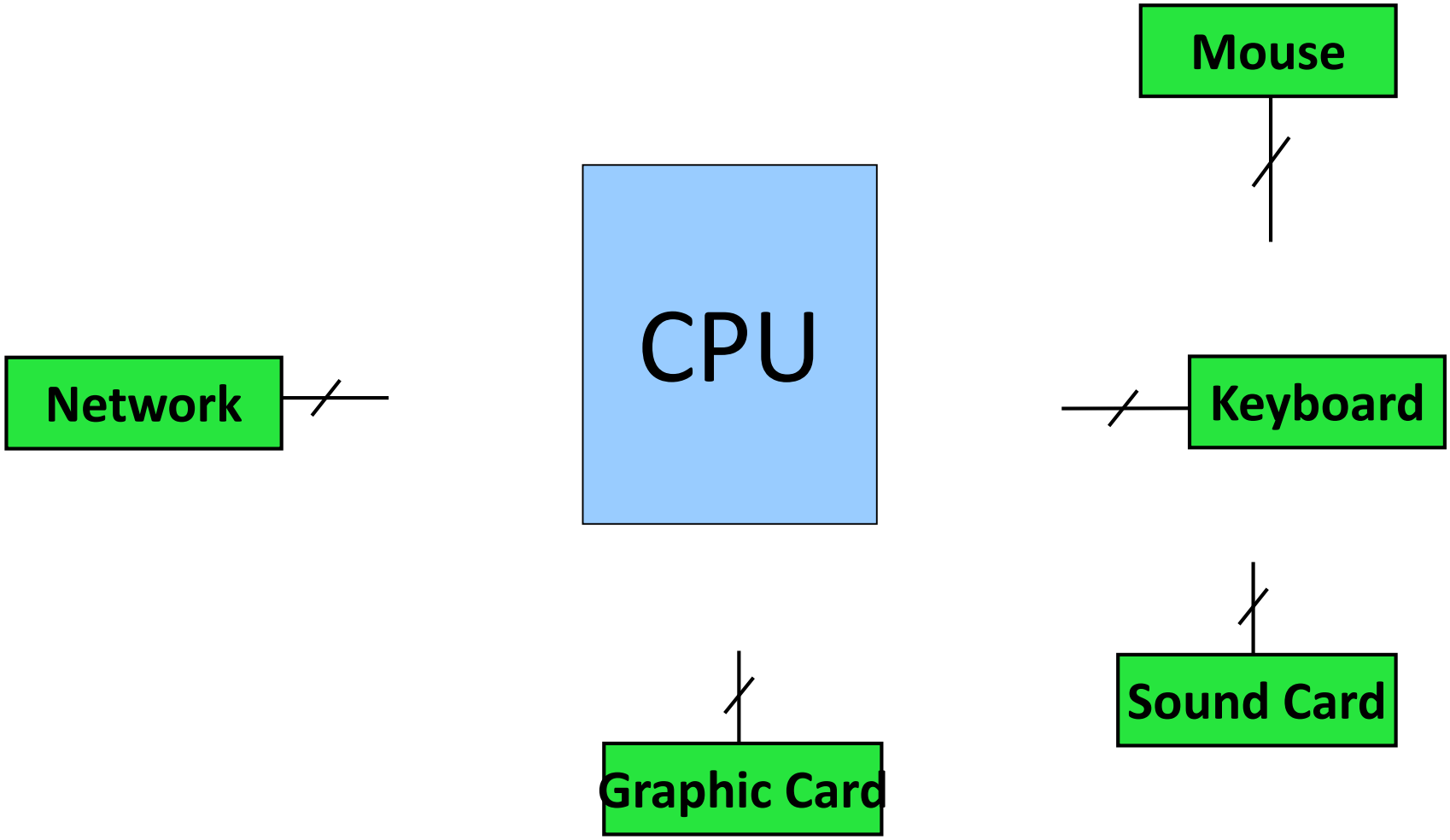
t_{ho}: Min data hold time after WE's rising edge

t_{RC} : Read Cycle Time

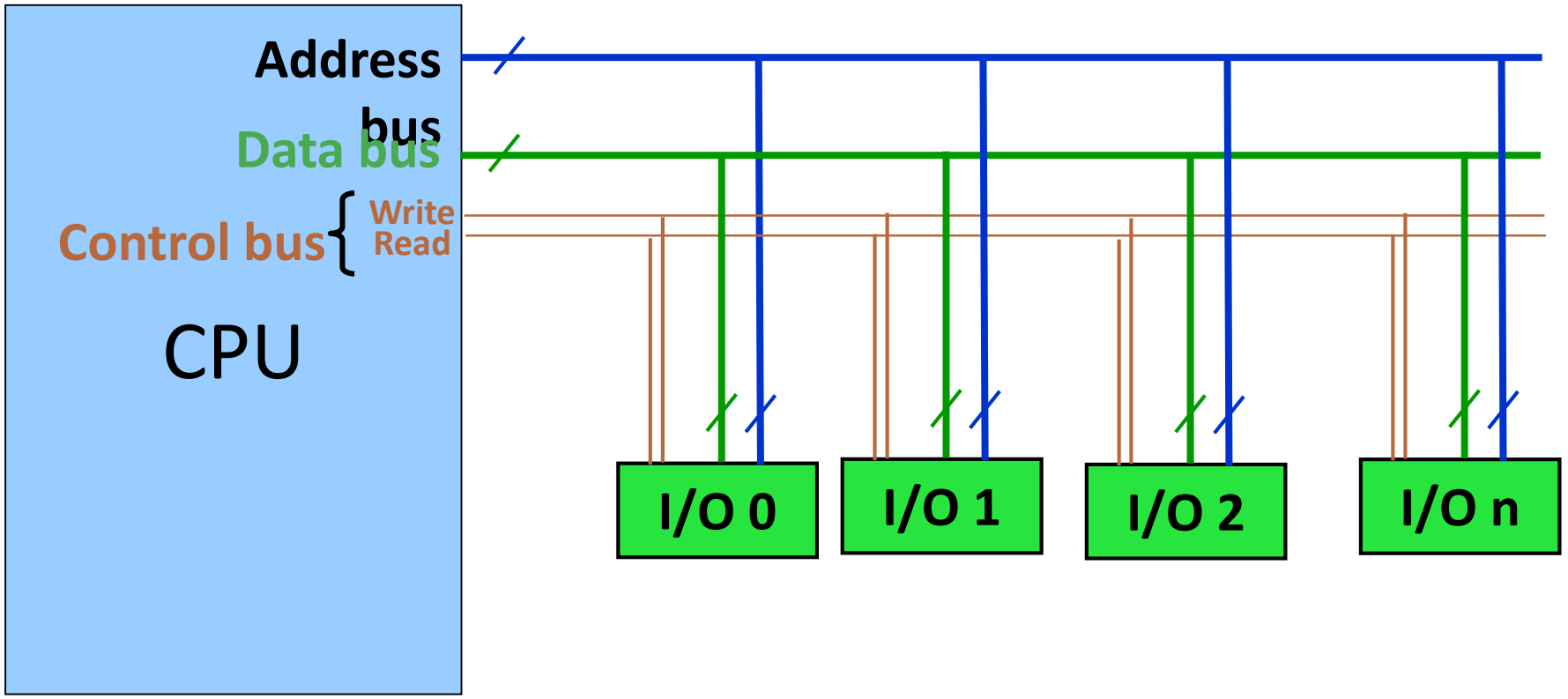
t_{AA} : Address Access Time

Connecting I/O devices to CPU

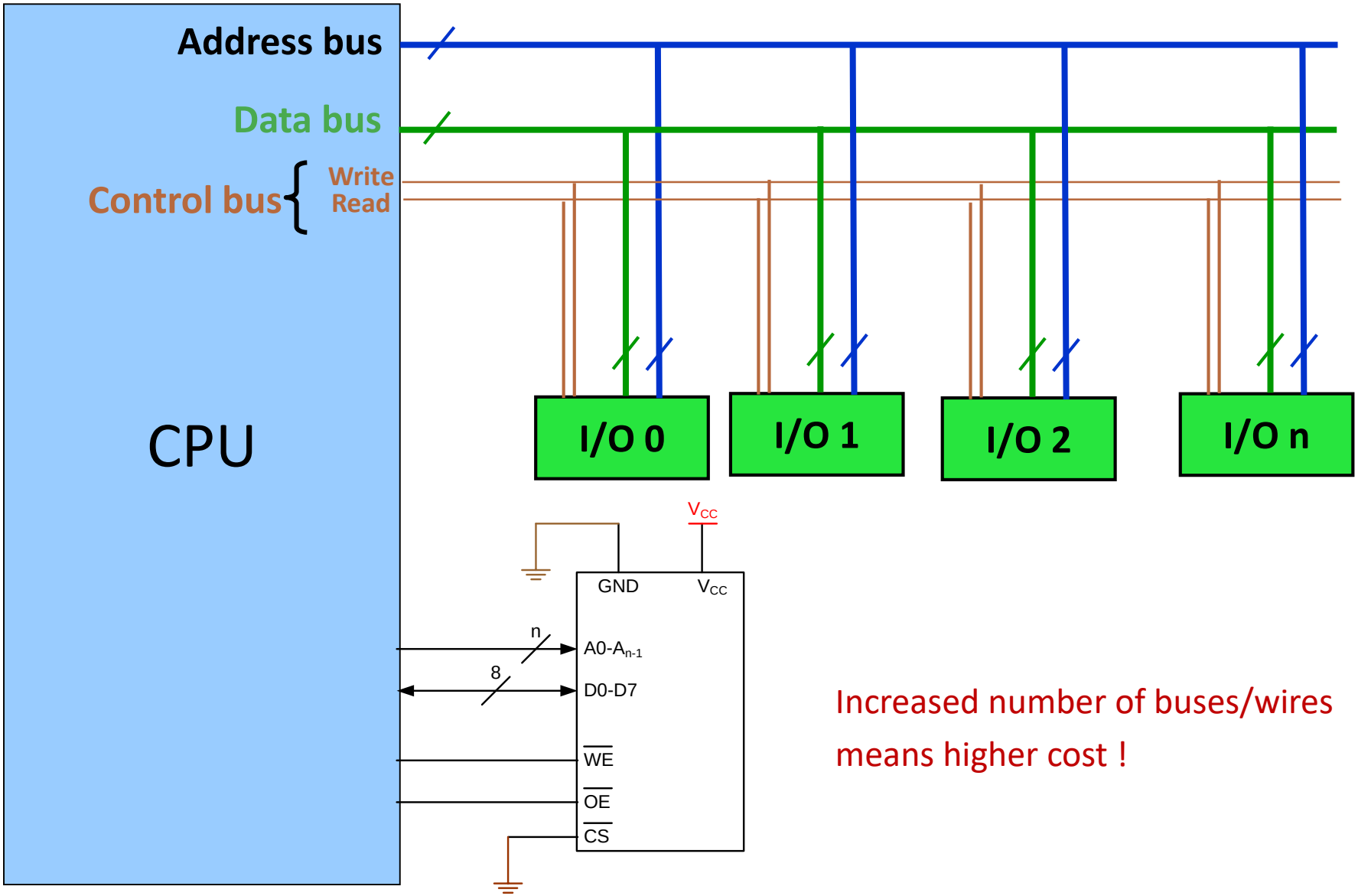
- CPU should have lots of pins!



Connecting I/O devices to CPU using a Shared Bus

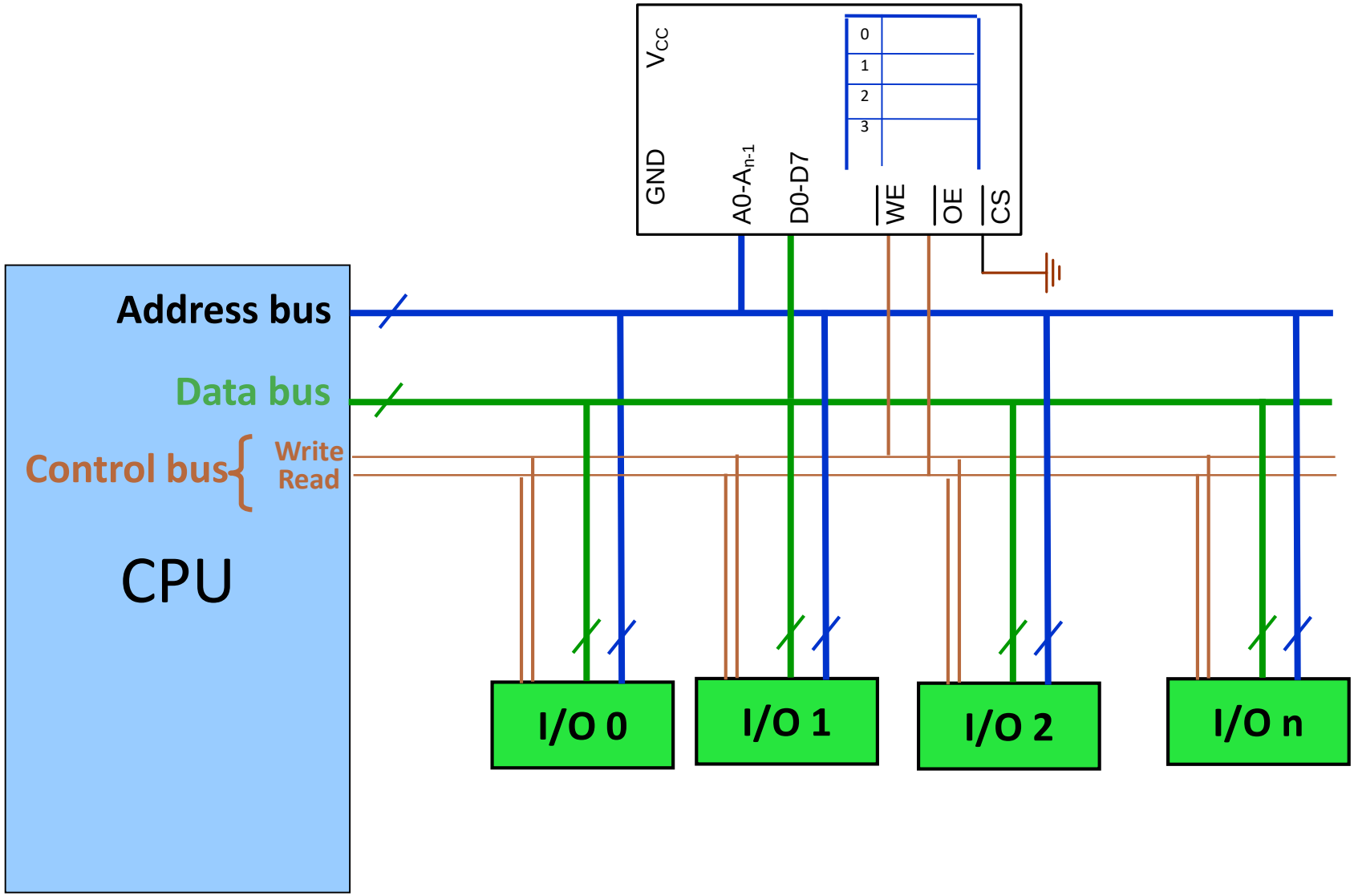


Connecting I/Os and Memory to CPU



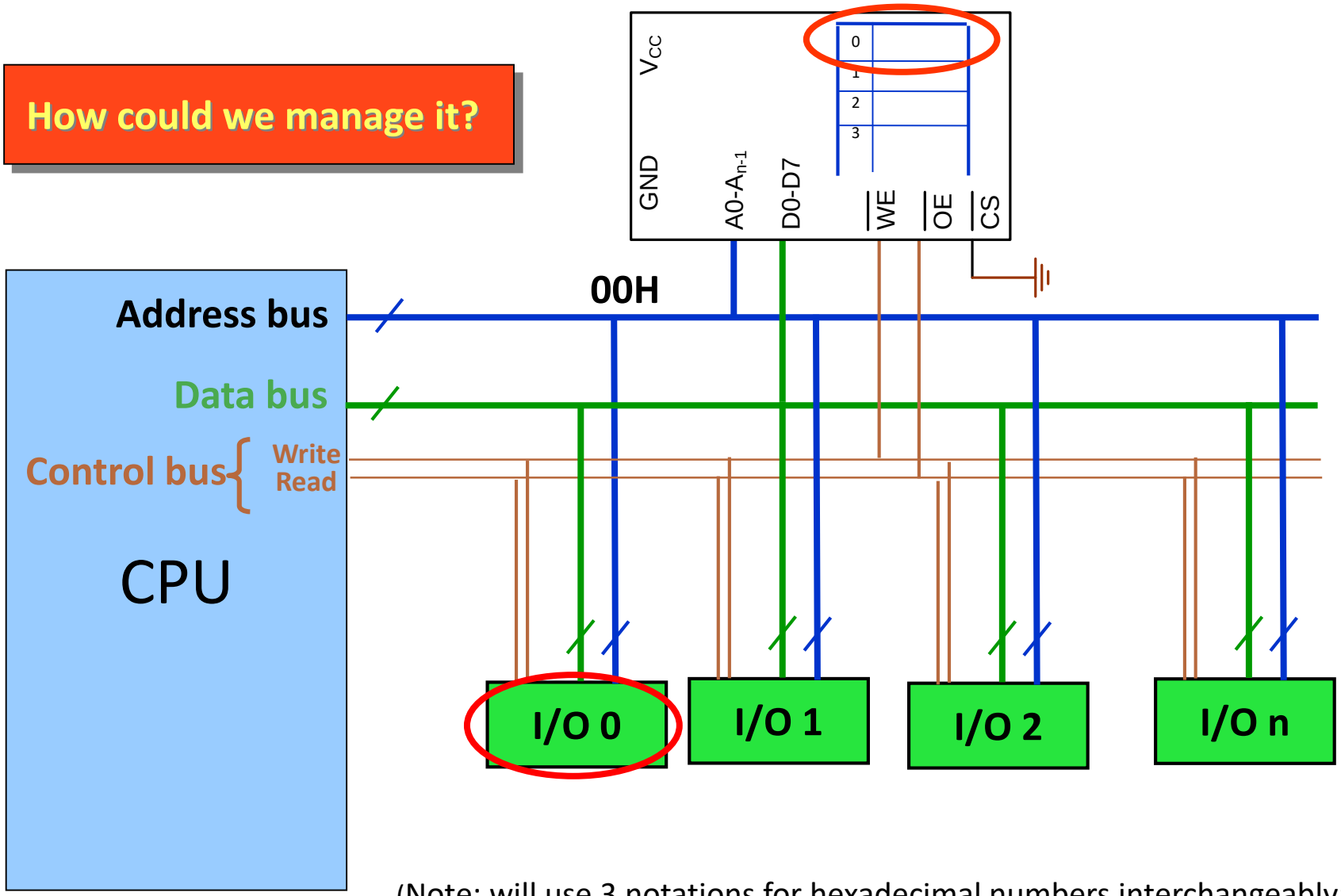
Increased number of buses/wires means higher cost !

Connecting I/Os and Memory to CPU Using a Shared Bus



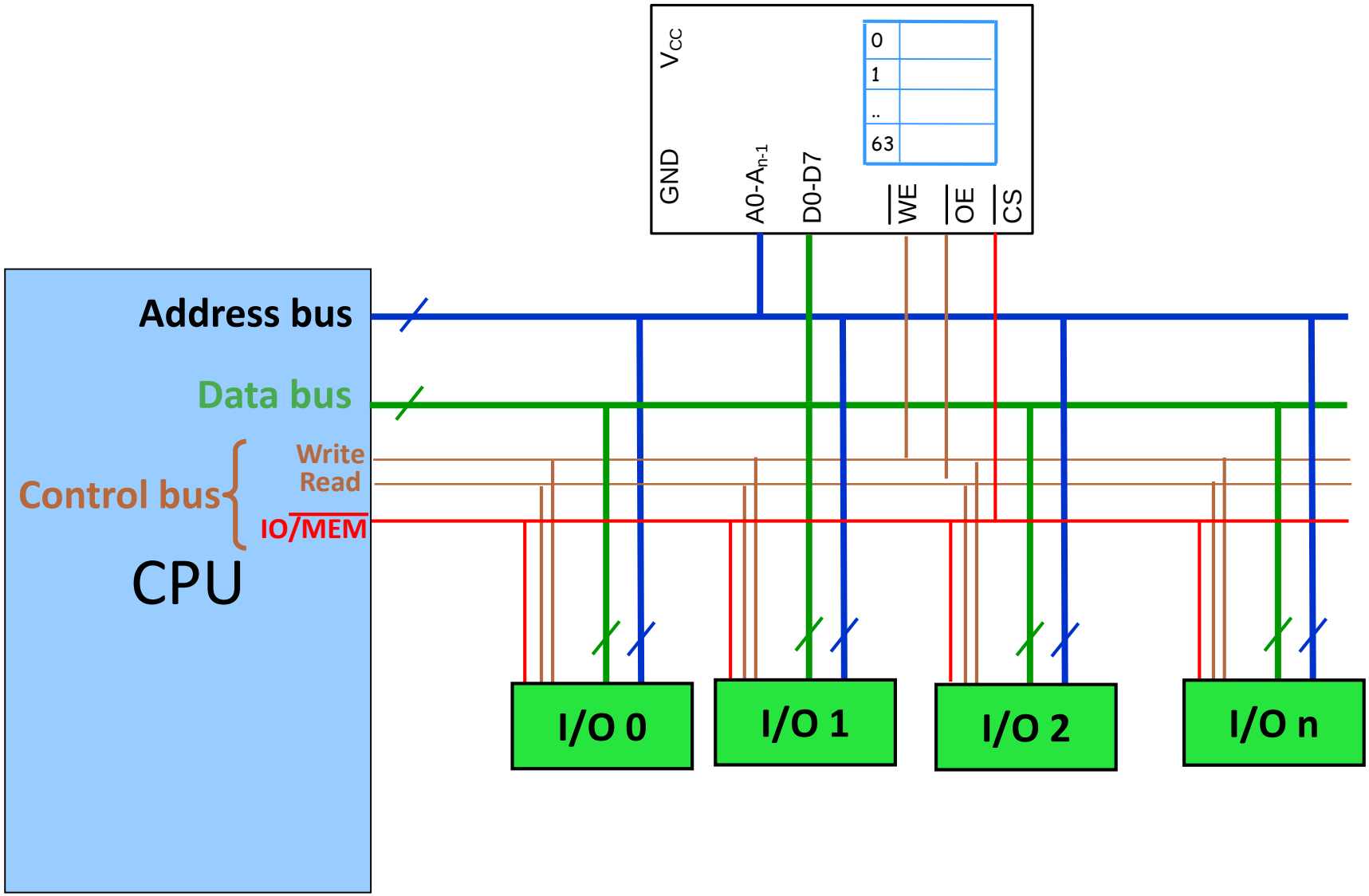
Connecting I/Os and Memory to CPU Using a Shared Bus

How could we manage it?

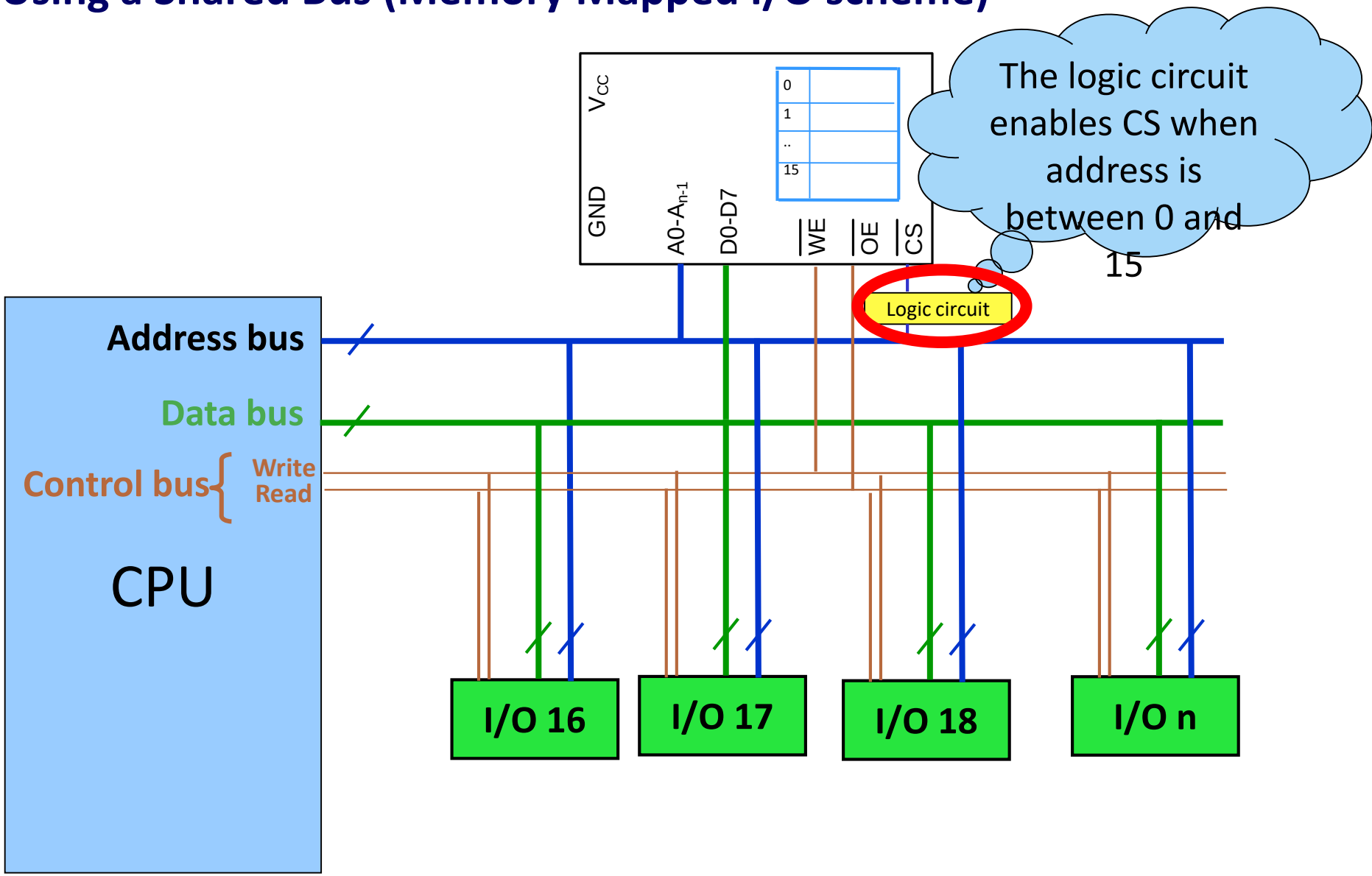


(Note: will use 3 notations for hexadecimal numbers interchangeably:
$Number, Number**H**, **0x**Number)

Connecting I/Os and Memory to CPU Using a Shared Bus (Peripheral I/O scheme)

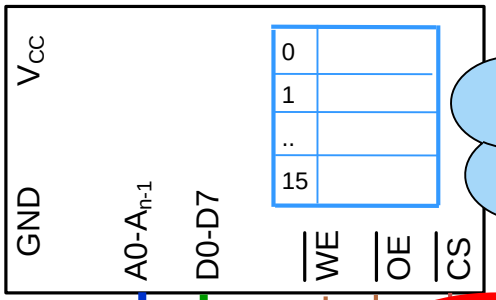


Connecting I/Os and Memory to CPU
Using a Shared Bus (Memory Mapped I/O scheme)



Connecting I/Os and Memory to CPU Using a Shared Bus (Memory Mapped I/O scheme)

How to design the logic circuit?



The logic circuit enables CS when address is between 0 and 15

Logic circuit

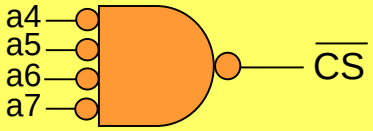
Address bus 8

Co

Solution

1. Write the address range in binary
2. Separate the fixed part of address
3. Using a NAND, design a logic circuit whose output activates when the fixed address is given to it.

	a7	a6	a5	a4	a3	a2	a1	a0
From address 0 ➔	0	0	0	0	0	0	0	0
To address15 ➔	0	0	0	0	1	1	1	1



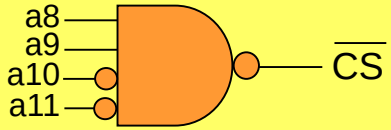
Another Example for Address Decoder

- Design an address decoder for address of 300H to 3FFH (for a 12-bit address bus)

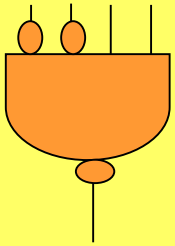
Solution

1. Write the address range in binary
2. Separate the fixed part of address
3. Design the logic circuit.

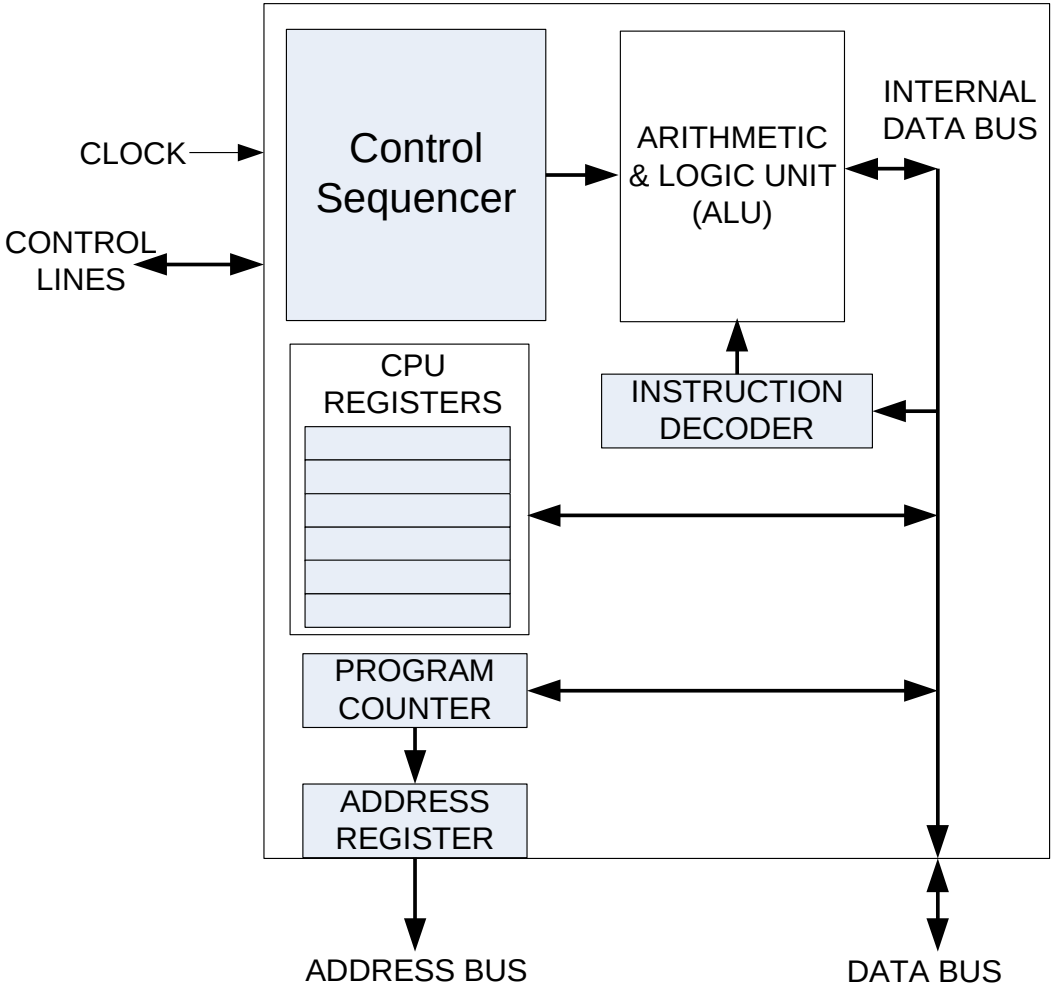
	a11	a10	a9	a8	a7	a6	a5	a4	a3	a2	a1	a0
From address 300H ➔	0	0	1	1	0	0	0	0	0	0	0	0
To address 3FFH ➔	0	0	1	1	1	1	1	1	1	1	1	1



An easy way of designing



Inside the CPU



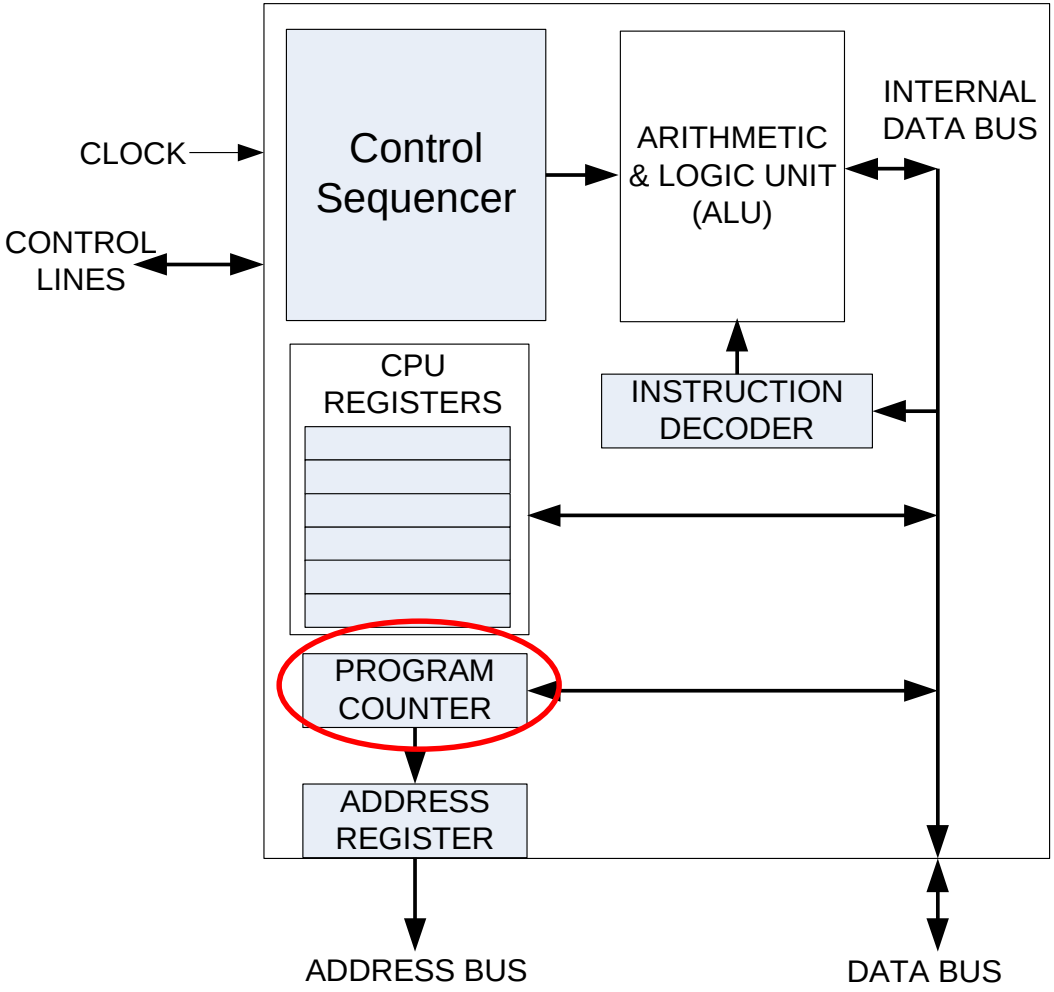
The main functions of the CPU are:

- Data transfer
- Arithmetic and logic operations
- Decision making (instruction flow control)

<u>Control Sequencer:</u>
Controls the operations in the CPU

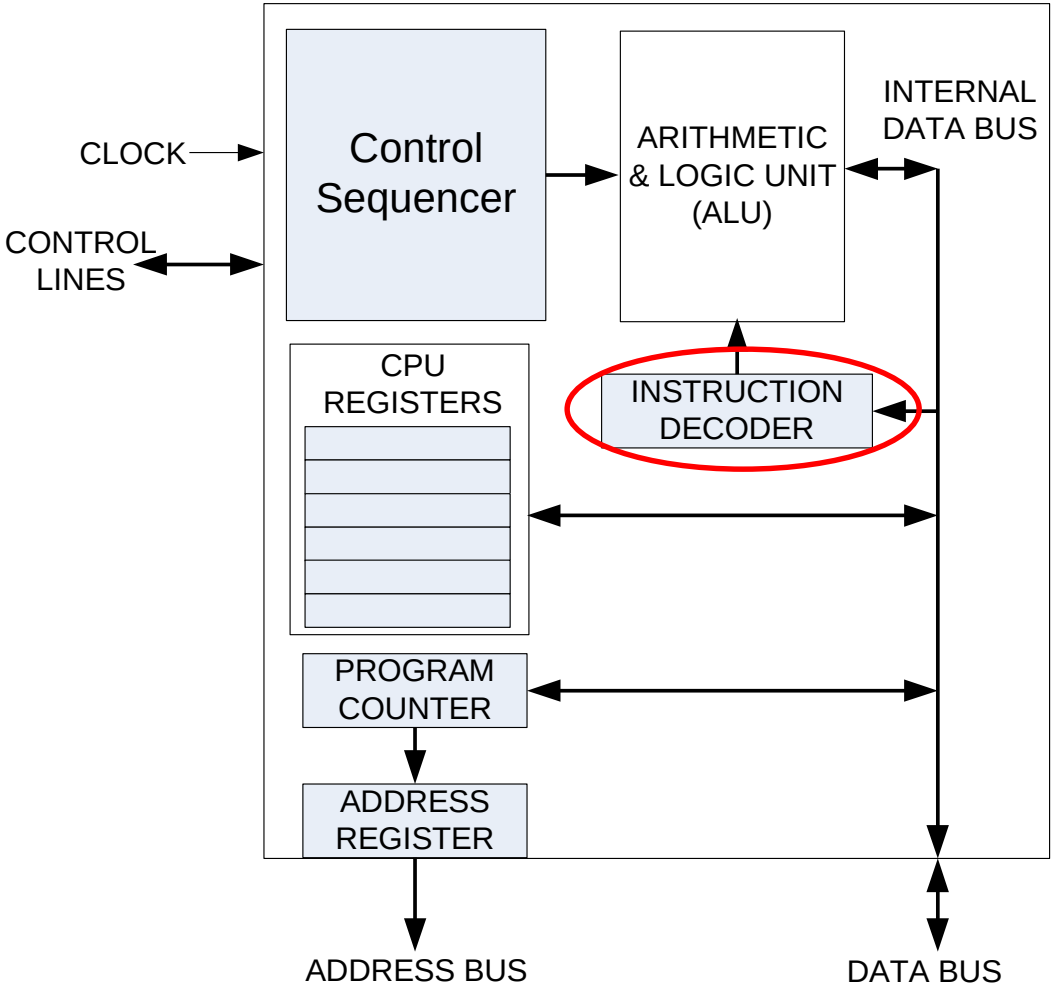
<u>Register Array:</u>
Temporary storage (faster than memory)

von Neumann Fetch/(Decode)/Execute Cycle



1. CPU keeps track of the location of the next instruction or data byte (to fetch) through the Program Counter (PC).

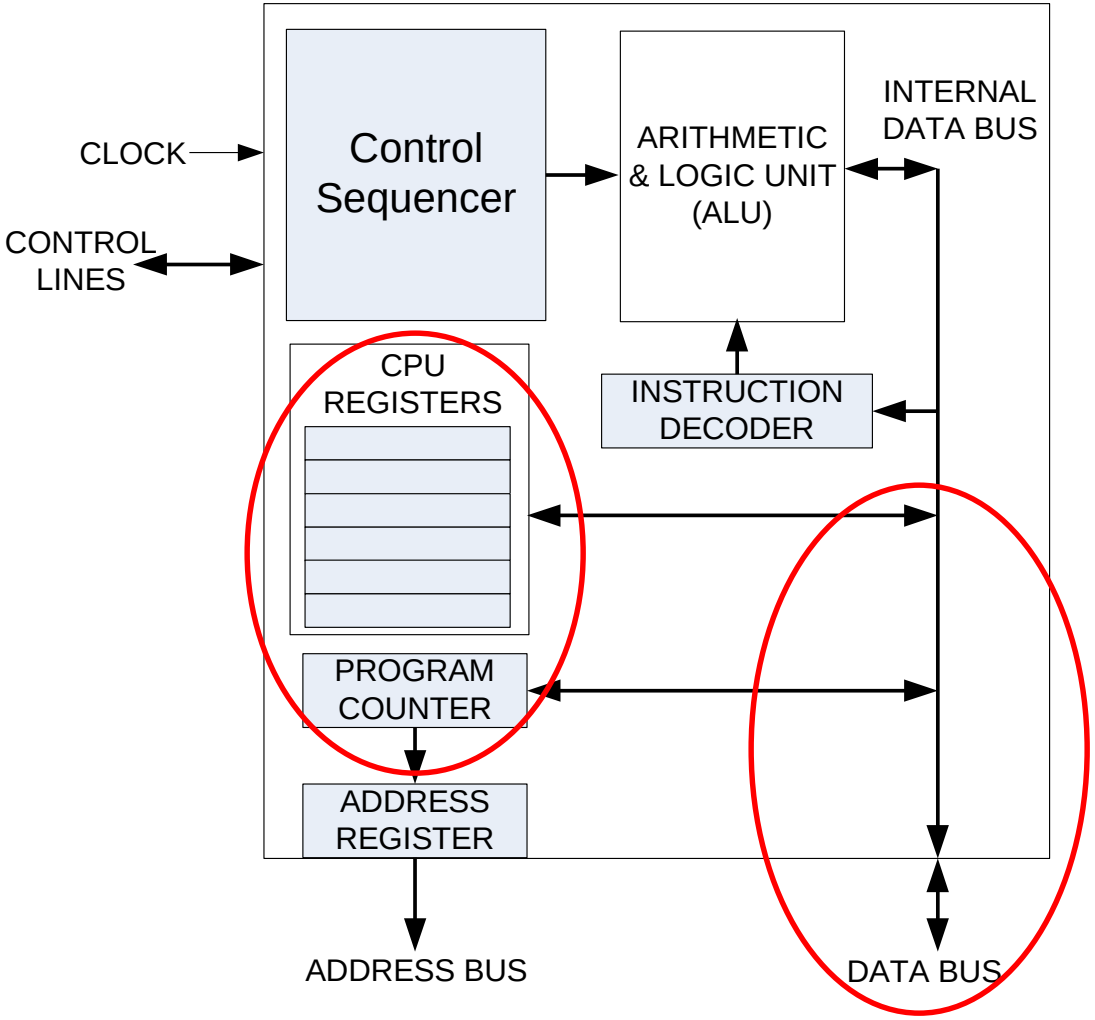
von Neumann Fetch/(Decode)/Execute Cycle



1. CPU keeps track of the location of the next instruction or data byte (to fetch) through the Program Counter (PC).

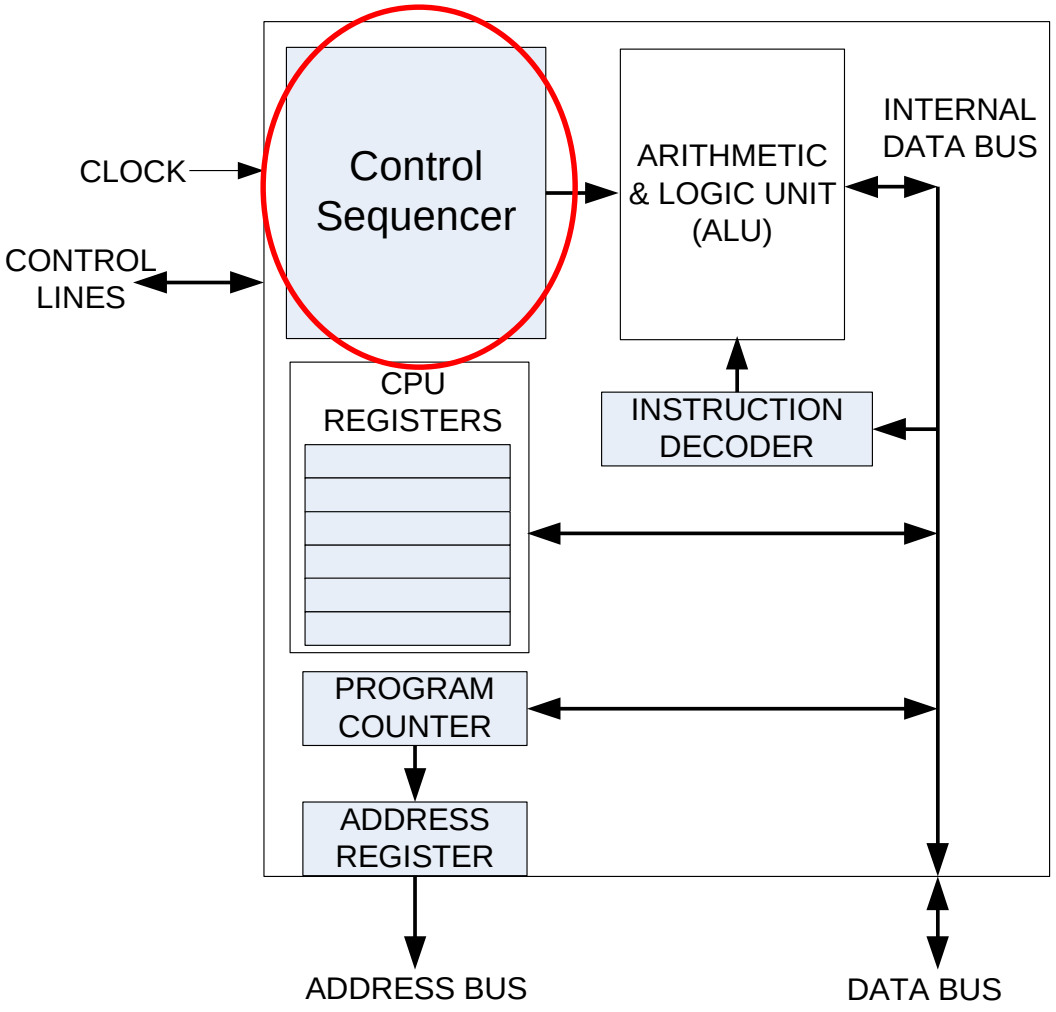
2. Instruction Decoder decodes the received data and configures the Arithmetic Logic Unit (ALU) to process it. Arithmetic or logic operations execute in the ALU.

von Neumann Fetch/(Decode)/Execute Cycle



1. CPU keeps track of the location of the next instruction or data byte (to fetch) through the Program Counter (PC).
2. Instruction Decoder decodes the received data and configures the Arithmetic Logic Unit (ALU) to process it. Arithmetic or logic operations execute in the ALU.
3. After the execution, the relevant bits are set, and data gets stored to a register or memory location if applicable.

von Neumann Fetch/(Decode)/Execute Cycle



1. CPU keeps track of the location of the next instruction or data byte (to fetch) through the Program Counter (PC).
2. Instruction Decoder decodes the received data and configures the Arithmetic Logic Unit (ALU) to process it. Arithmetic or logic operations execute in the ALU.
3. After the execution, the relevant bits are set, and data gets stored to a register or memory location if applicable.
4. The Control Sequencer ensures the Fetch/Decode/Execute cycle repeats until the last instruction is detected.

Instruction Set Architecture

- An interface to allow easy communication between the programmer and the hardware.
- Instruction Set: It is a group of instructions that can be presented to the computer.
- A typical instruction consists of two parts: Opcode and Operand.
  - Opcode or operational code is the instruction applied. It can be loading data, storing data etc.
  - Opcode encoding depends on the number of bits used
  - Example: For ARM, all instructions are of 32-bit length, but only 8 bits (bit 20 to 28) are used to encode the instruction. Hence a total of 28 = 256 different instructions are possible. (THUMB2-some 16 bit)
- Operand is the memory register or data upon which instruction is applied.
  - ADD R3,R2,R1; R2+R1 -> R3

Addressing Modes

- Addressing modes are the manner how the data is accessed.
- Depending upon the type of instruction applied, addressing modes are of various types such as direct mode where straight data is accessed or indirect mode where the location of the data is operand.
- ADD R3,R2,R1; R2+R1 -> R3
- ADD R3,R3,#1; R3+ 1 -> R3

Registers

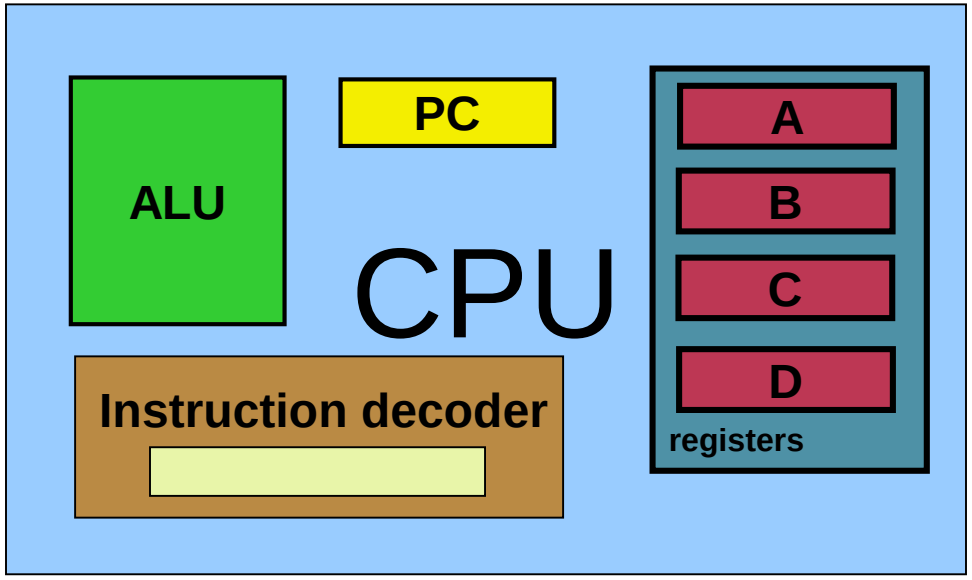
- The most fundamental storage area in the processor
  - is closely located to the processor
  - provides very fast access, operating at the same frequency as the processor clock
  - but is of limited quantity (typically less than 100)
- Most are of the general purpose type and can store any type of information:
  - data –e.g., timer value, constants
  - address –e.g., ASCII table, stack
- Some are reserved for specific purposes
  - program counter (PC in 8051)
  - program status word (PSW in 8051)

CISC vs RISC

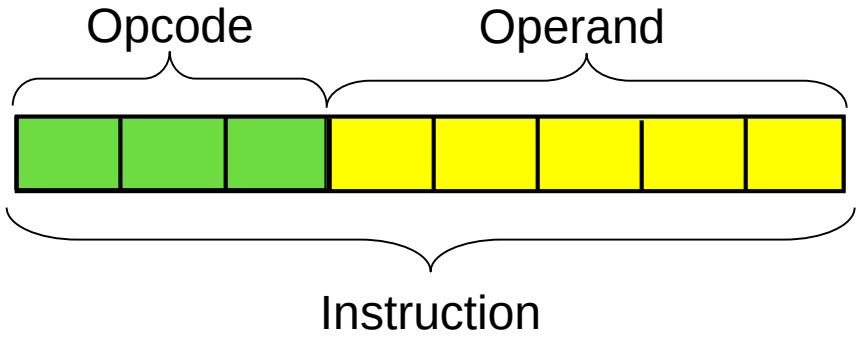
- CISC
  - Variable length instructions
  - Small number of registers
  - We can have memory operands in an arithmetic instruction
- RISC
  - Fixed length instructions
  - Large number of registers
  - Only Load and Store instructions are used to access memory operands
  - Because RISC requires register operands, data transfers are required before arithmetic operations

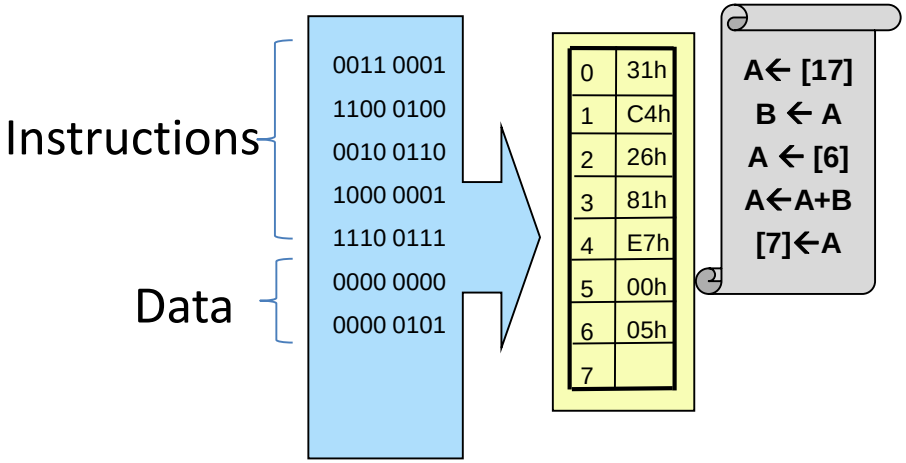
Example: Consider a Simple CPU

- PC (Program Counter)
- Instruction decoder
- ALU (Arithmetic Logic Unit)
- Registers



Instruction Decode

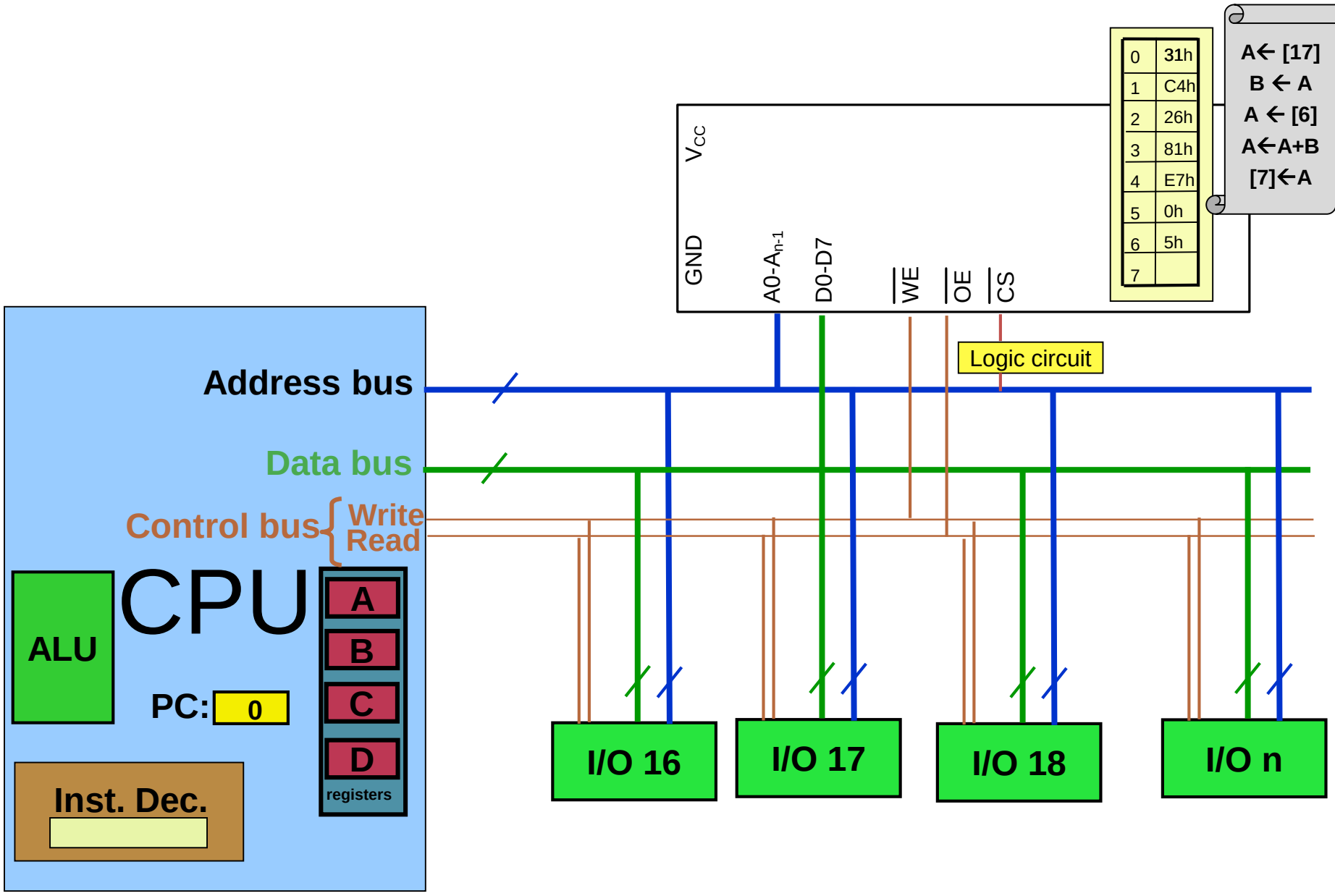




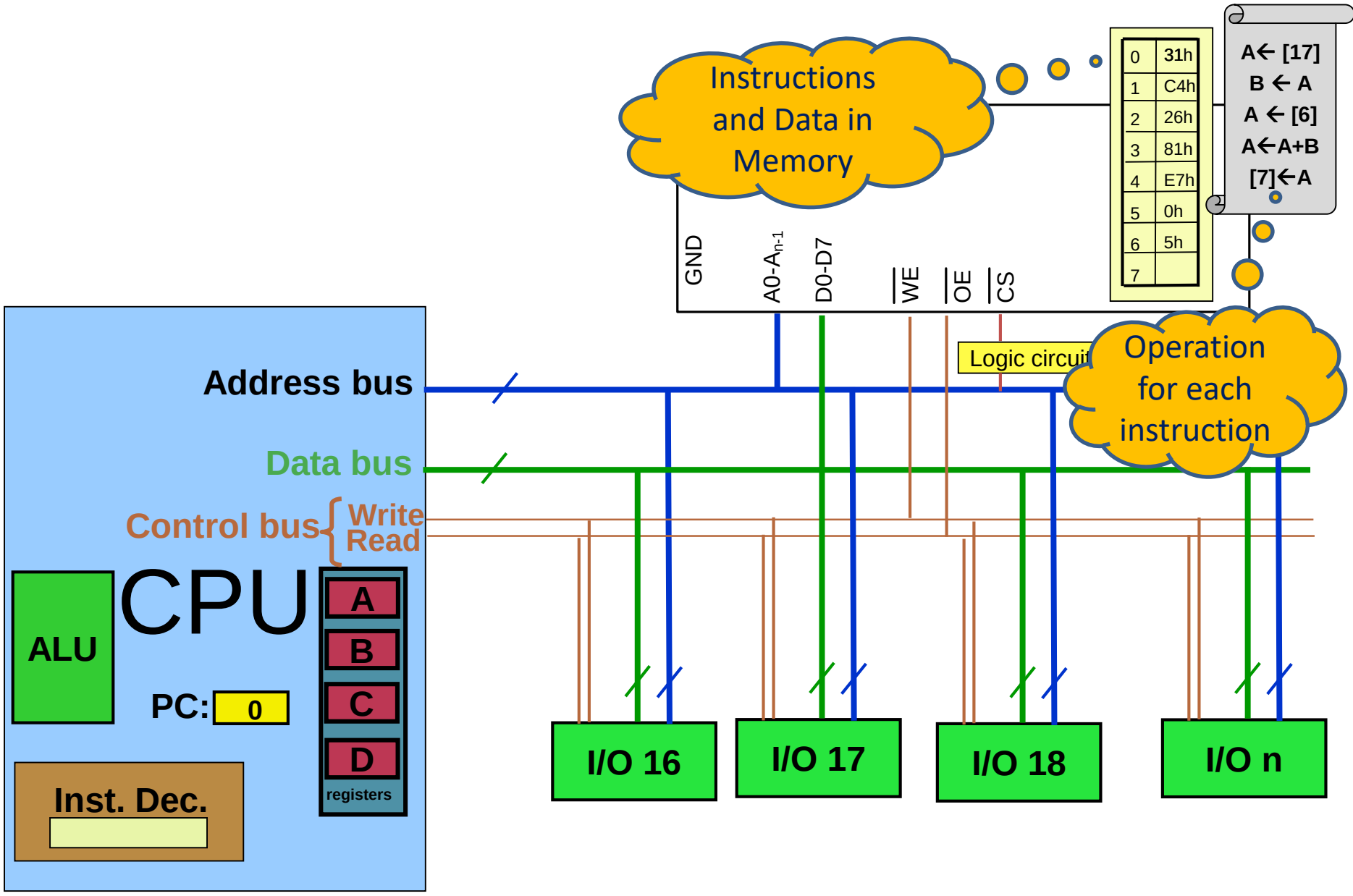
Operation Code	Meaning
000	A ← x
001	A ← [x]
010	A ← A – register (x)
011	A ← A + x
100	A ← A + register (x)
101	A ← A – x
110	Register (x_H) ← Register (x_L)
111	[x] ← A

Note: These are hypothetical instructions used for this example…

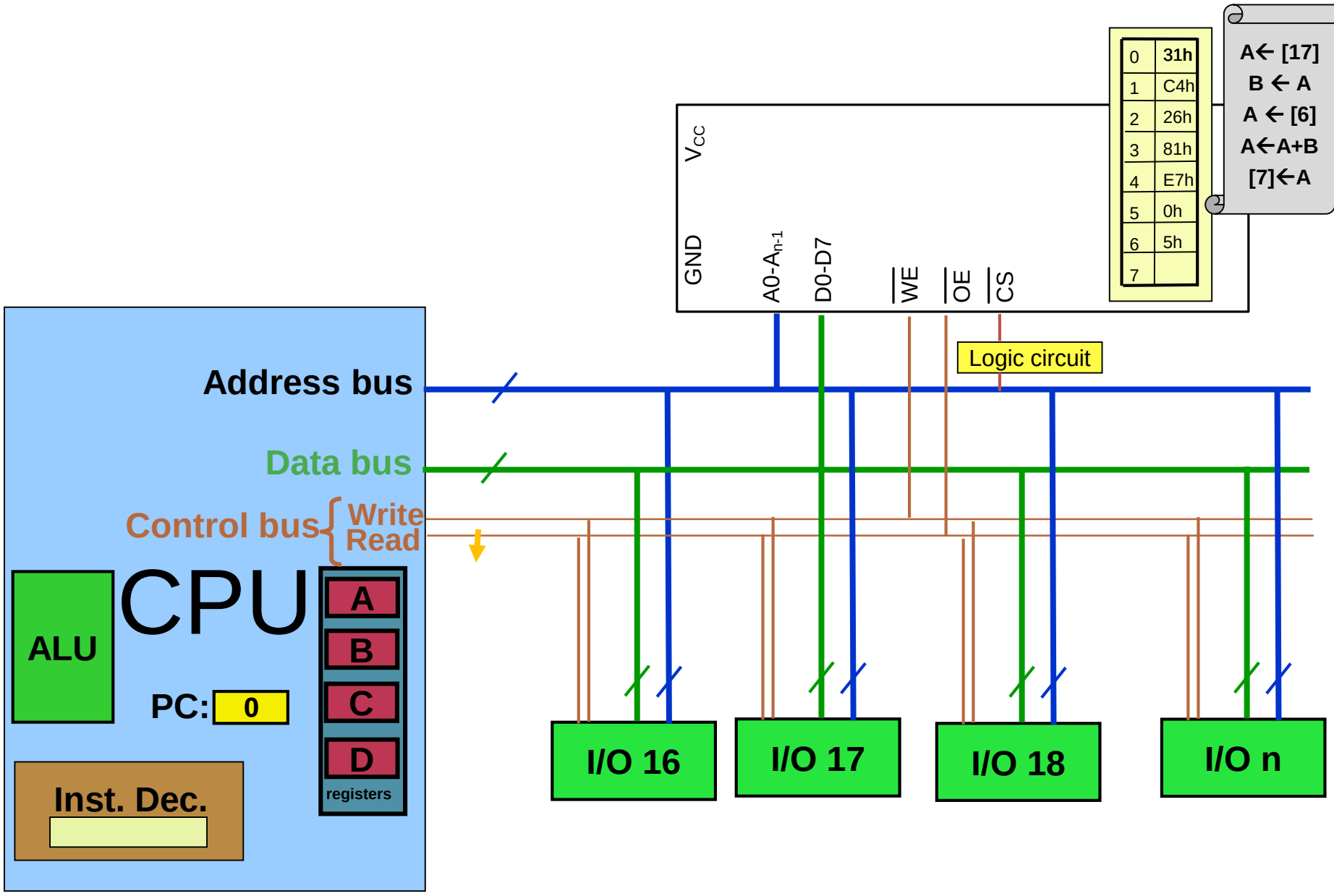
Instruction Fetch➔(Decode)➔Execute in CPU



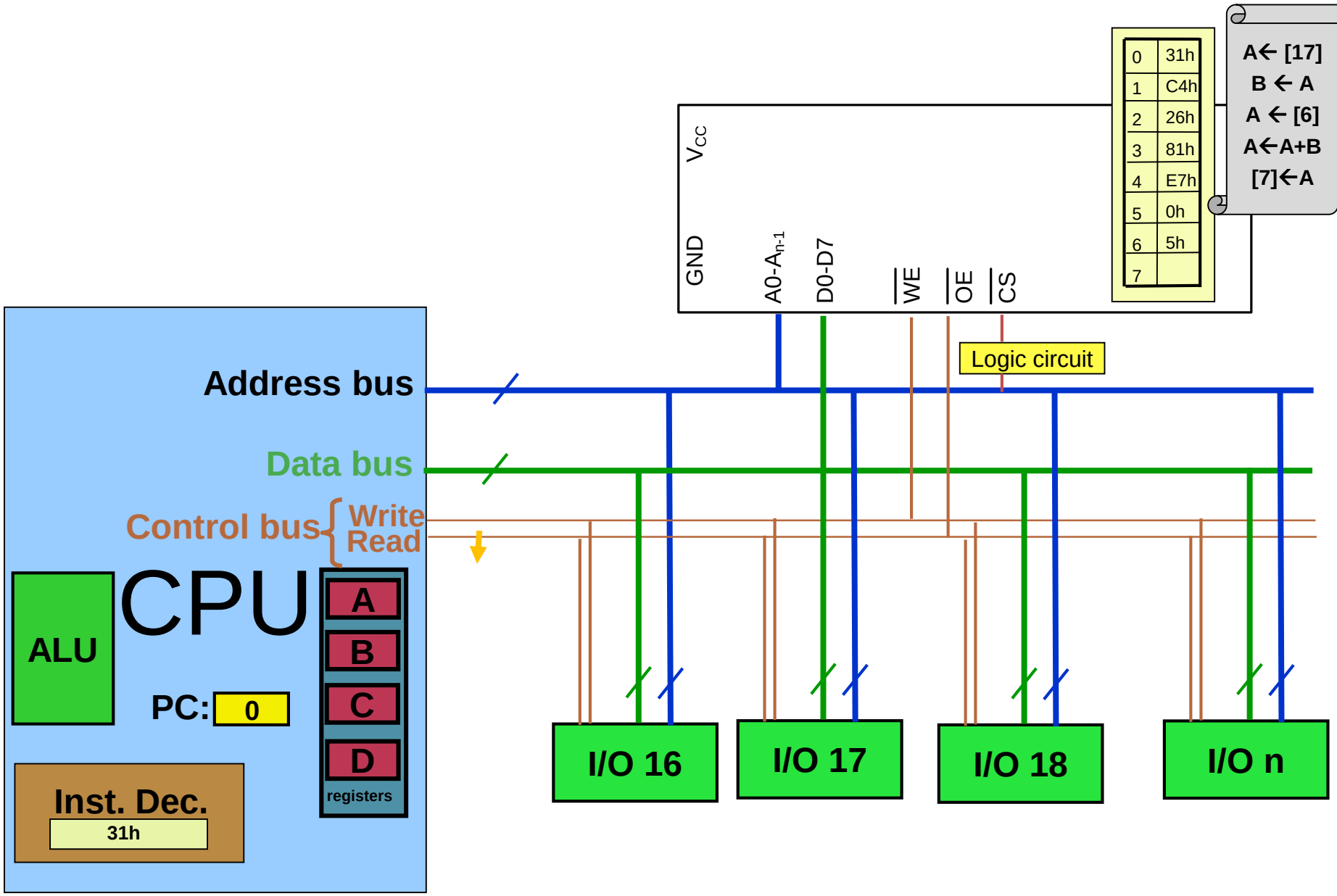
Instruction Fetch➔(Decode)➔Execute in CPU



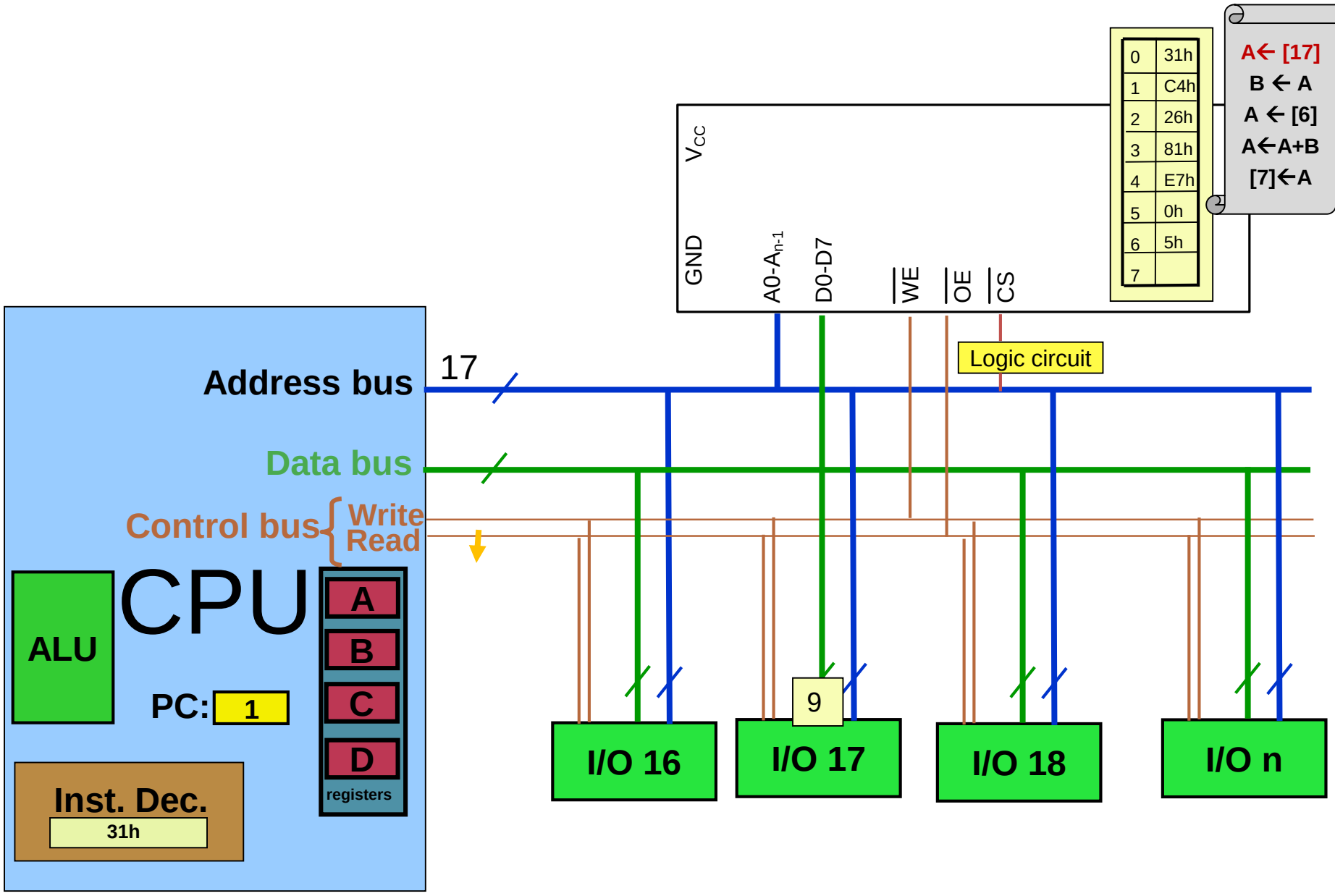
Instruction Fetch➔(Decode)➔Execute in CPU



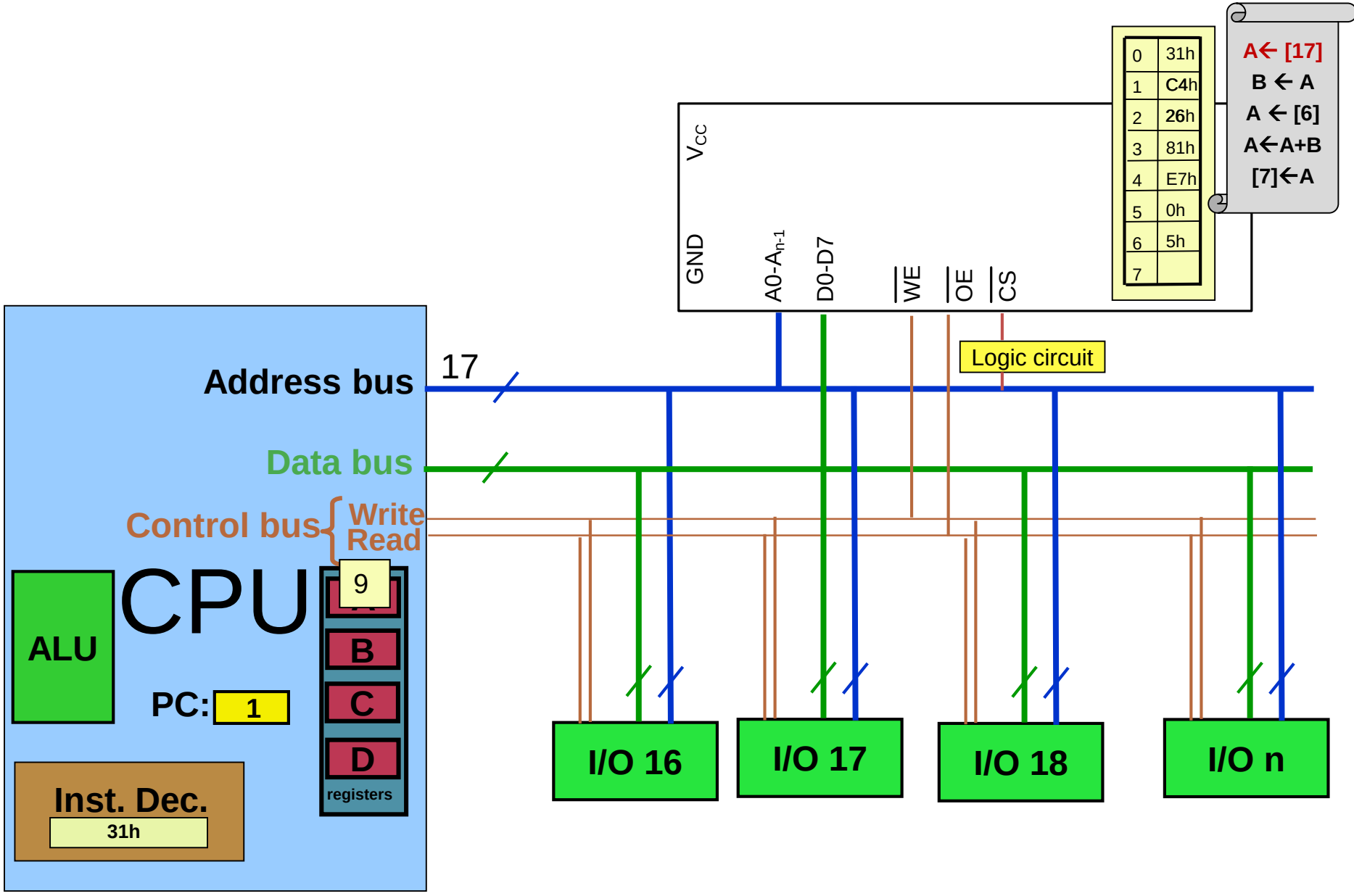
Instruction Fetch➔(Decode)➔Execute in CPU



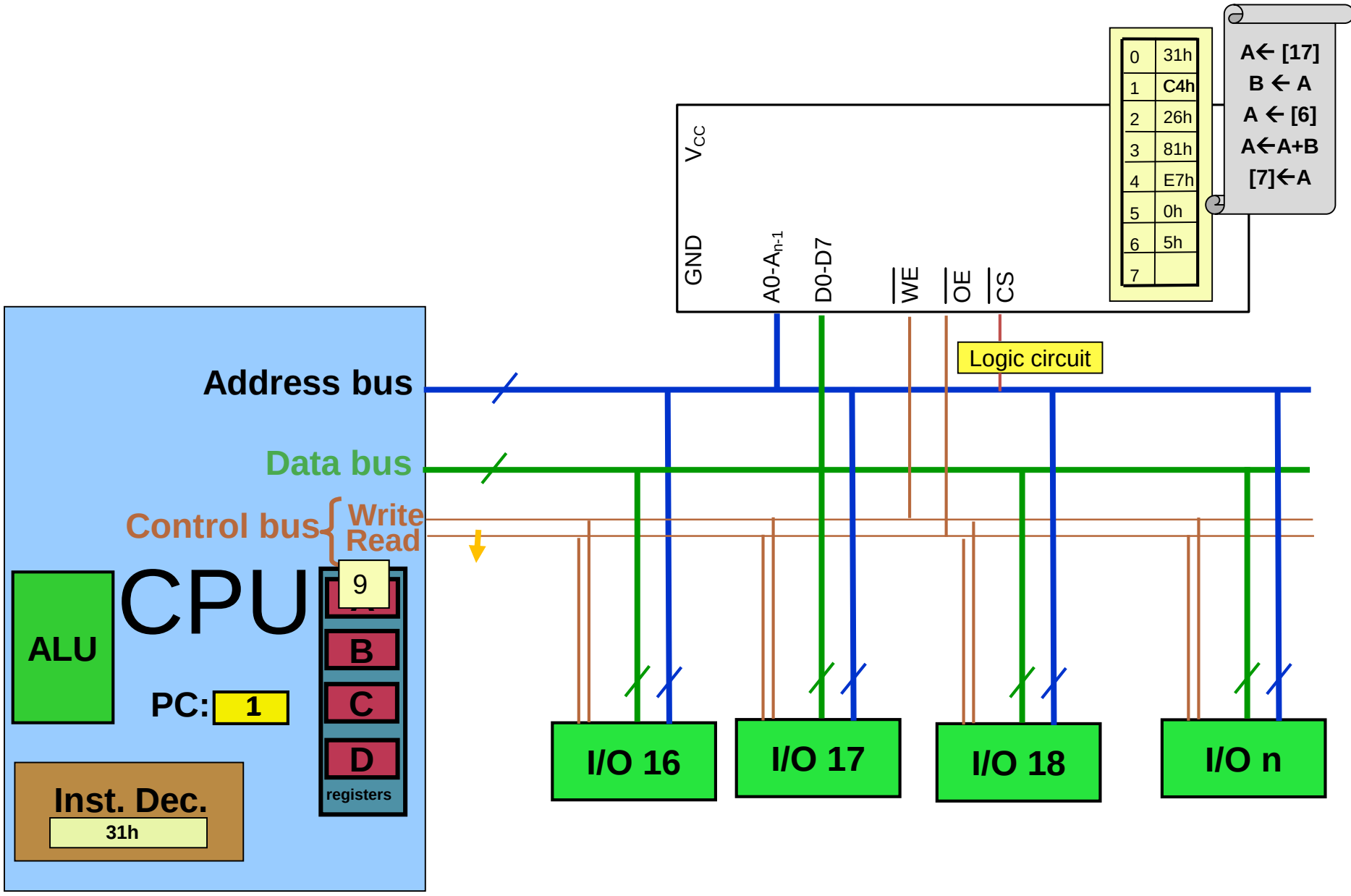
Instruction Fetch➔(Decode)➔Execute in CPU



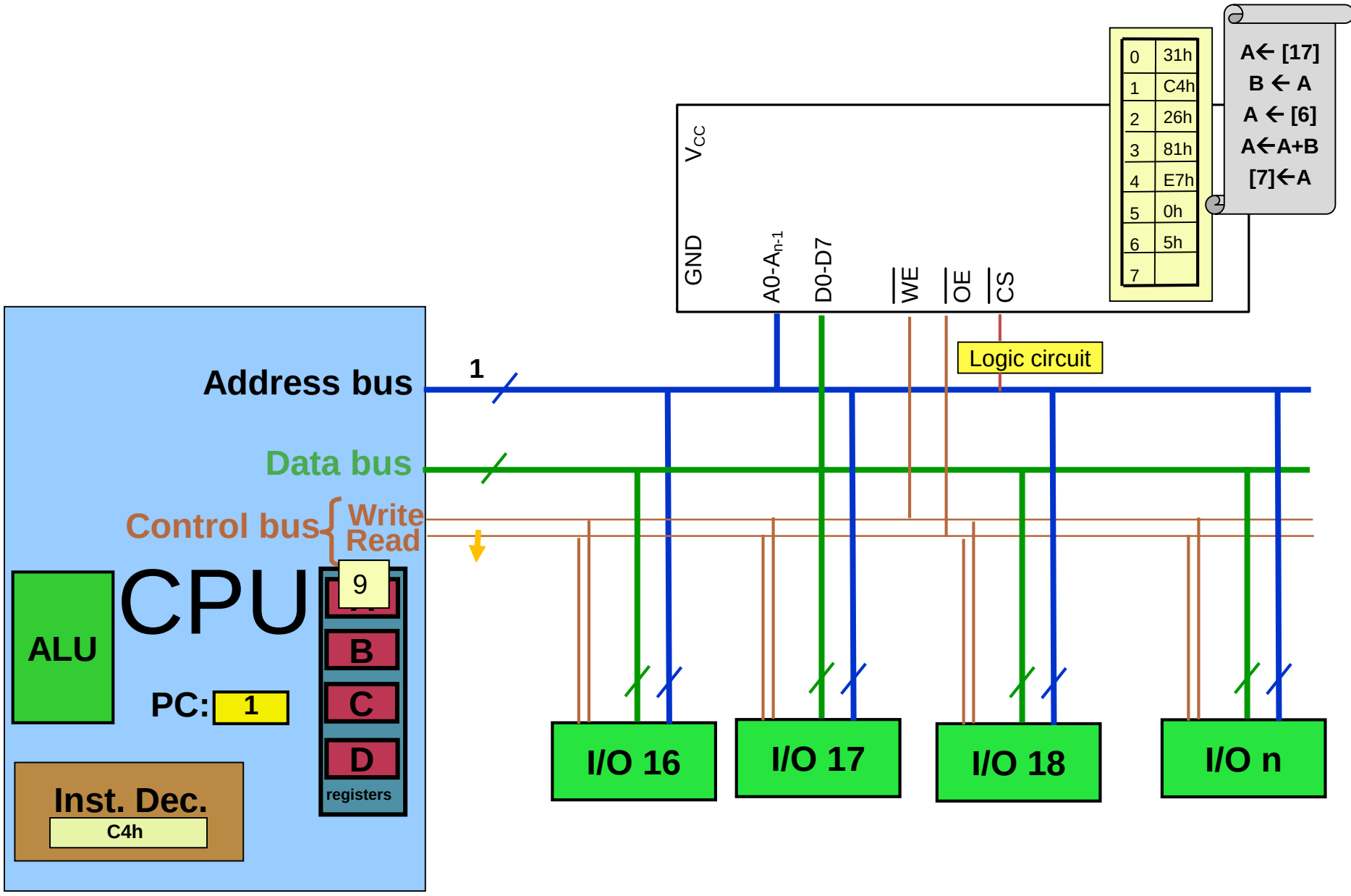
Instruction Fetch➔(Decode)➔Execute in CPU



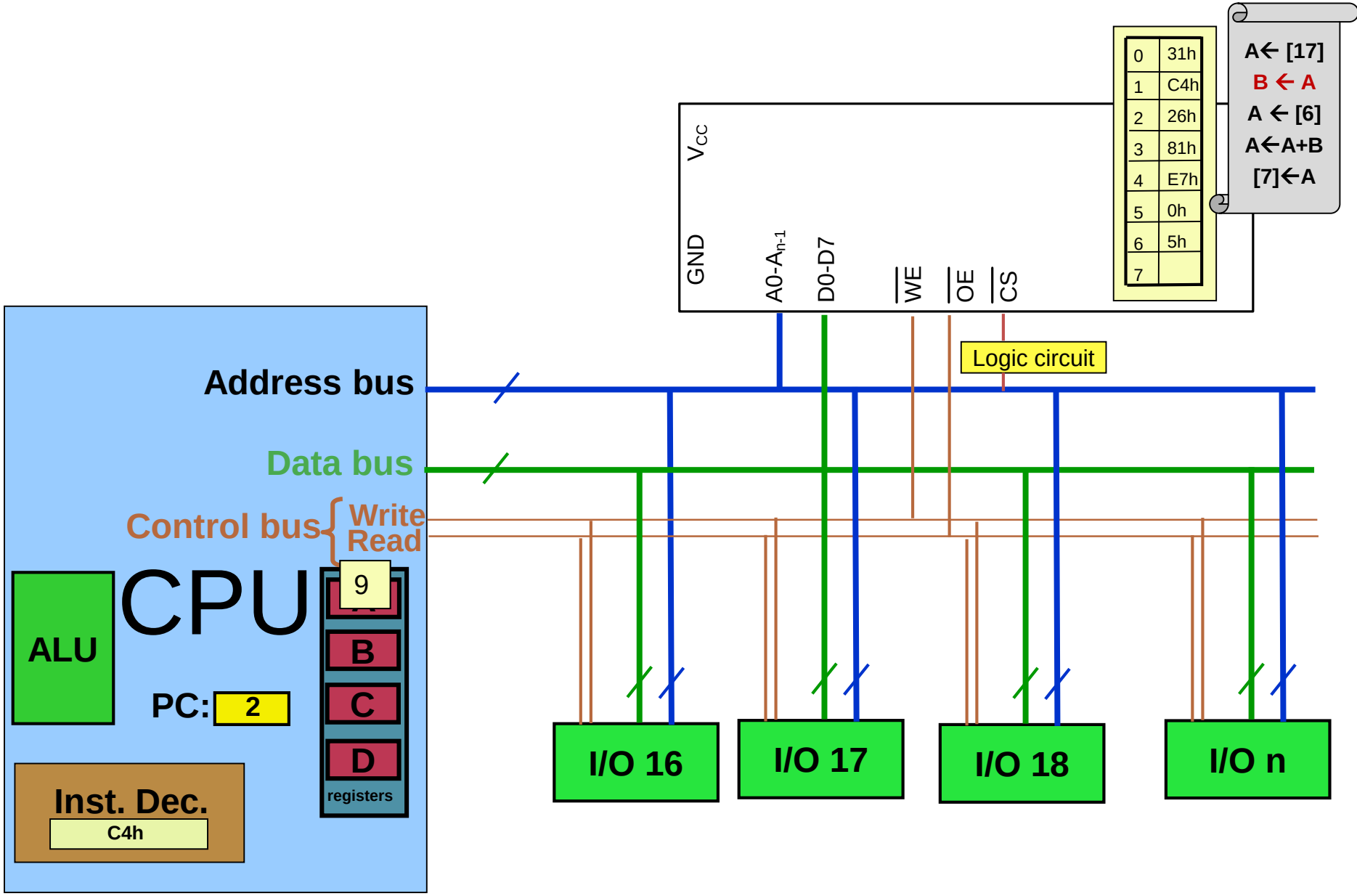
Instruction Fetch➔(Decode)➔Execute in CPU



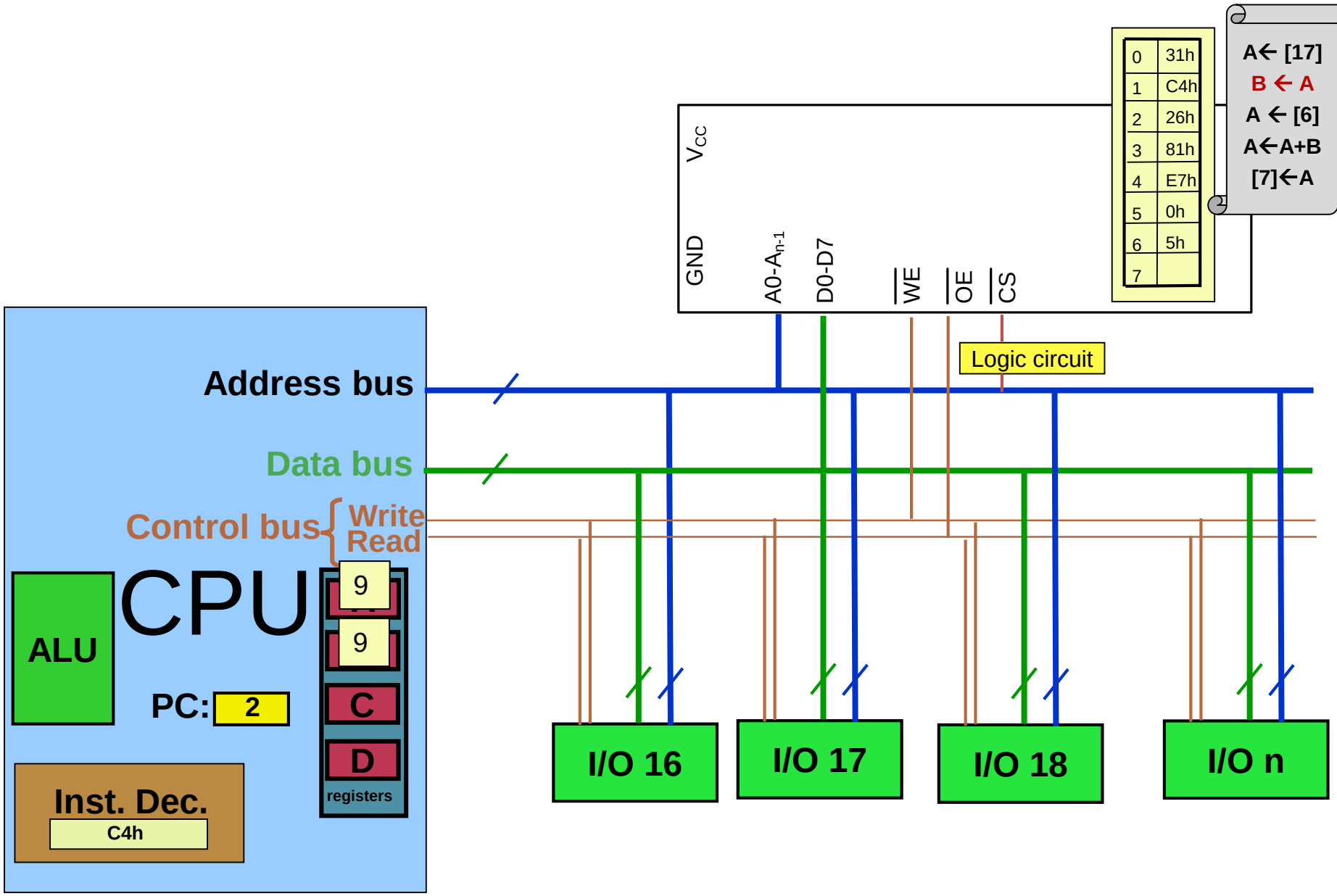
Instruction Fetch➔(Decode)➔Execute in CPU



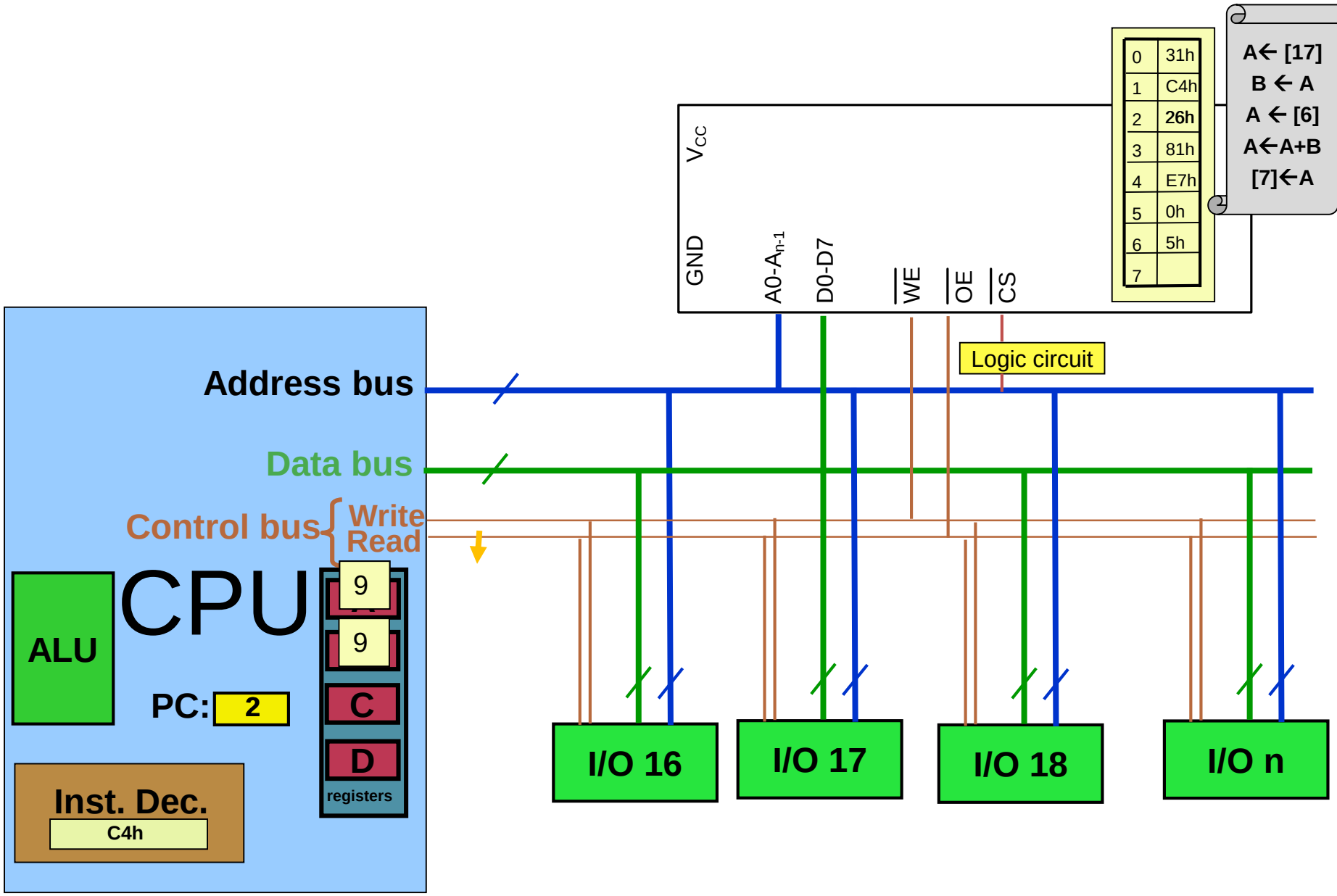
Instruction Fetch➔(Decode)➔Execute in CPU



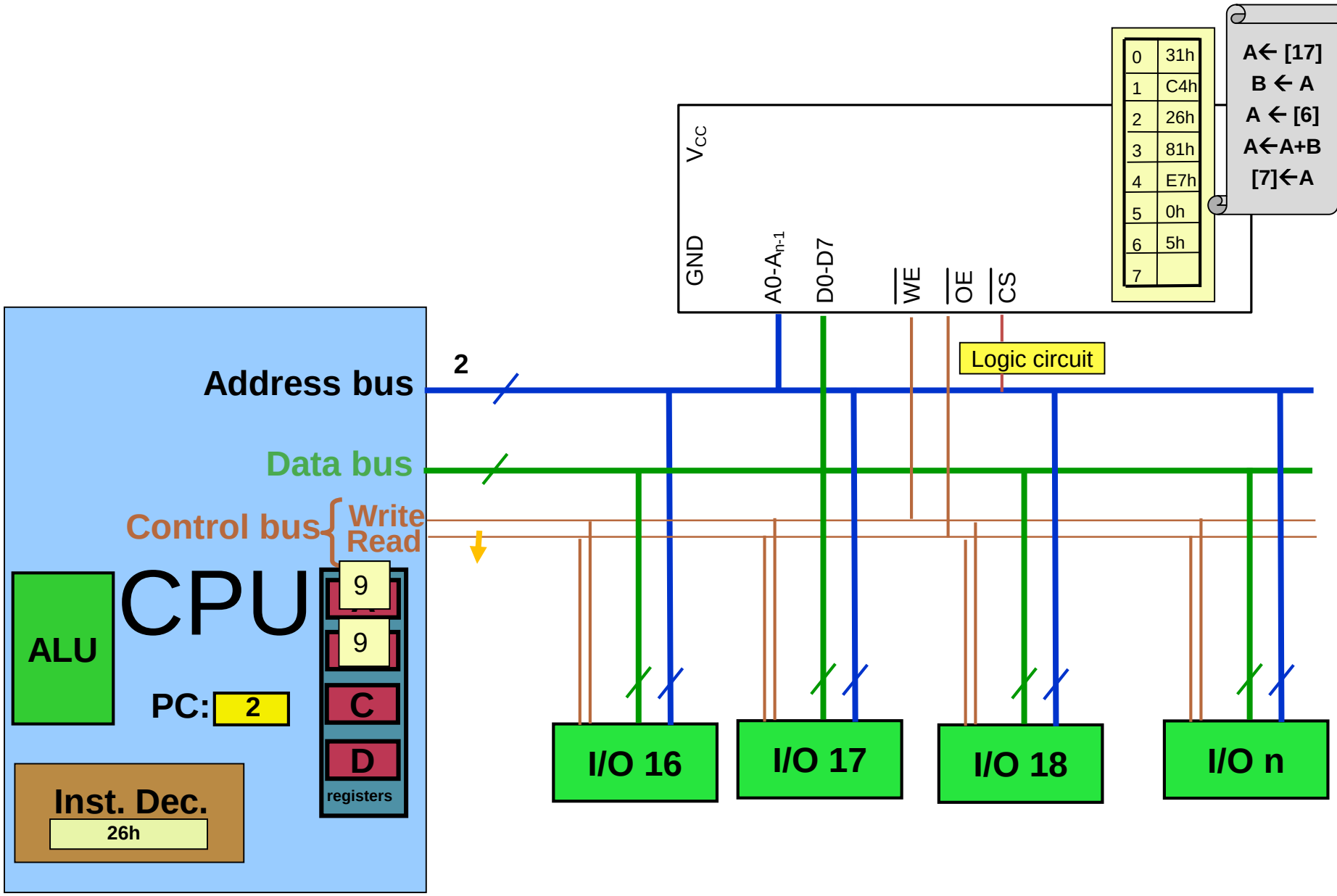
Instruction Fetch➔(Decode)➔Execute in CPU



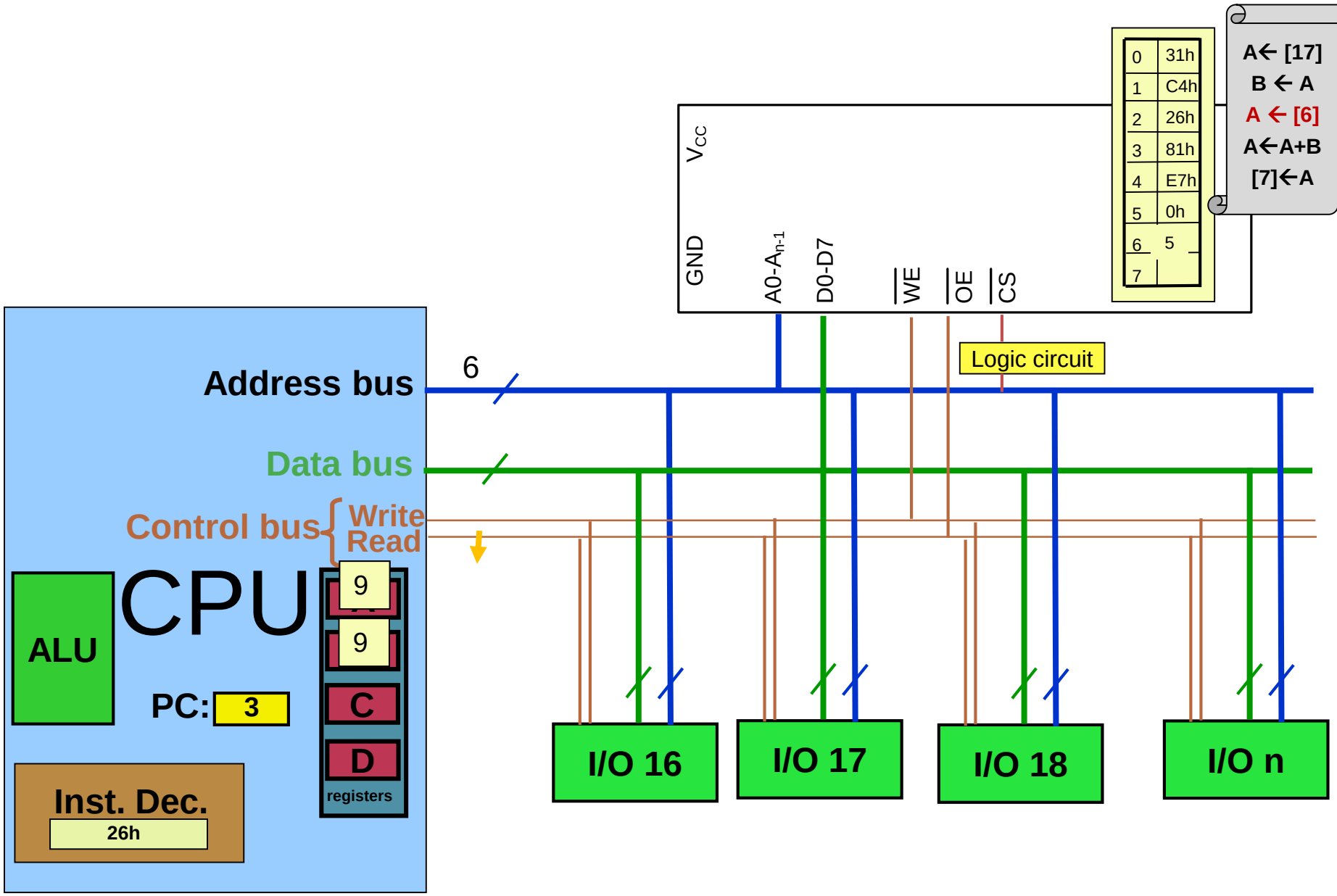
Instruction Fetch➔(Decode)➔Execute in CPU



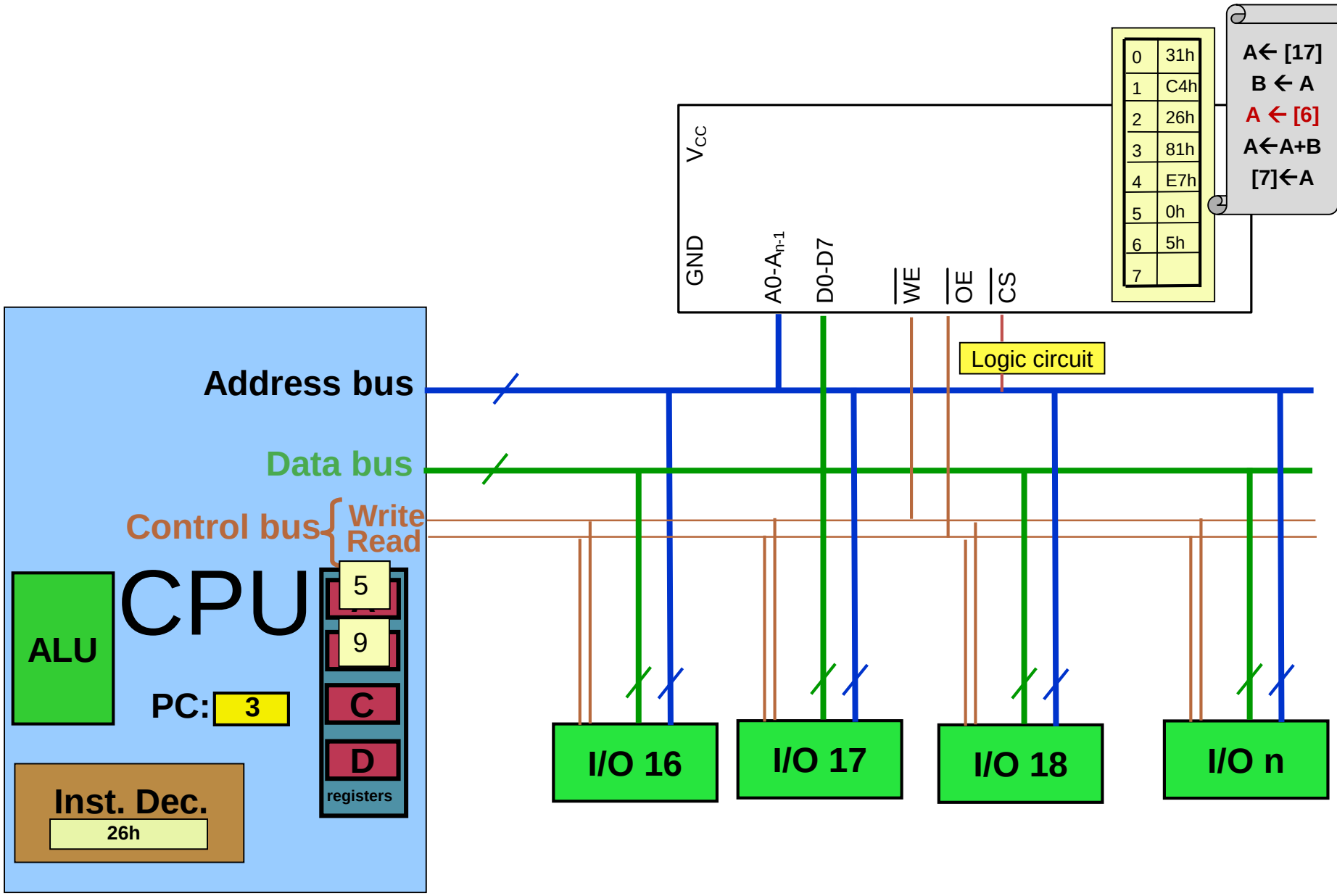
Instruction Fetch➔(Decode)➔Execute in CPU



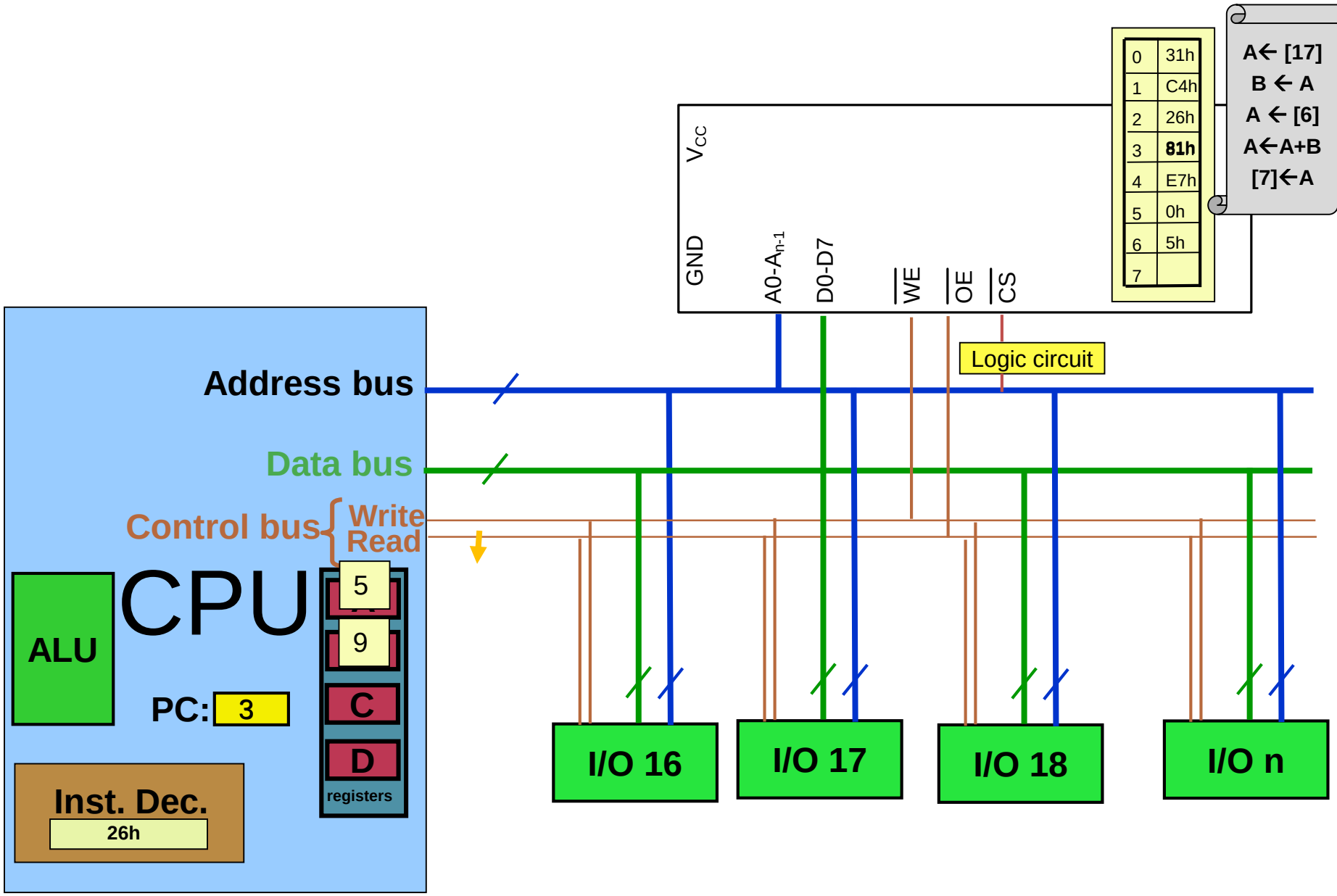
Instruction Fetch➔(Decode)➔Execute in CPU



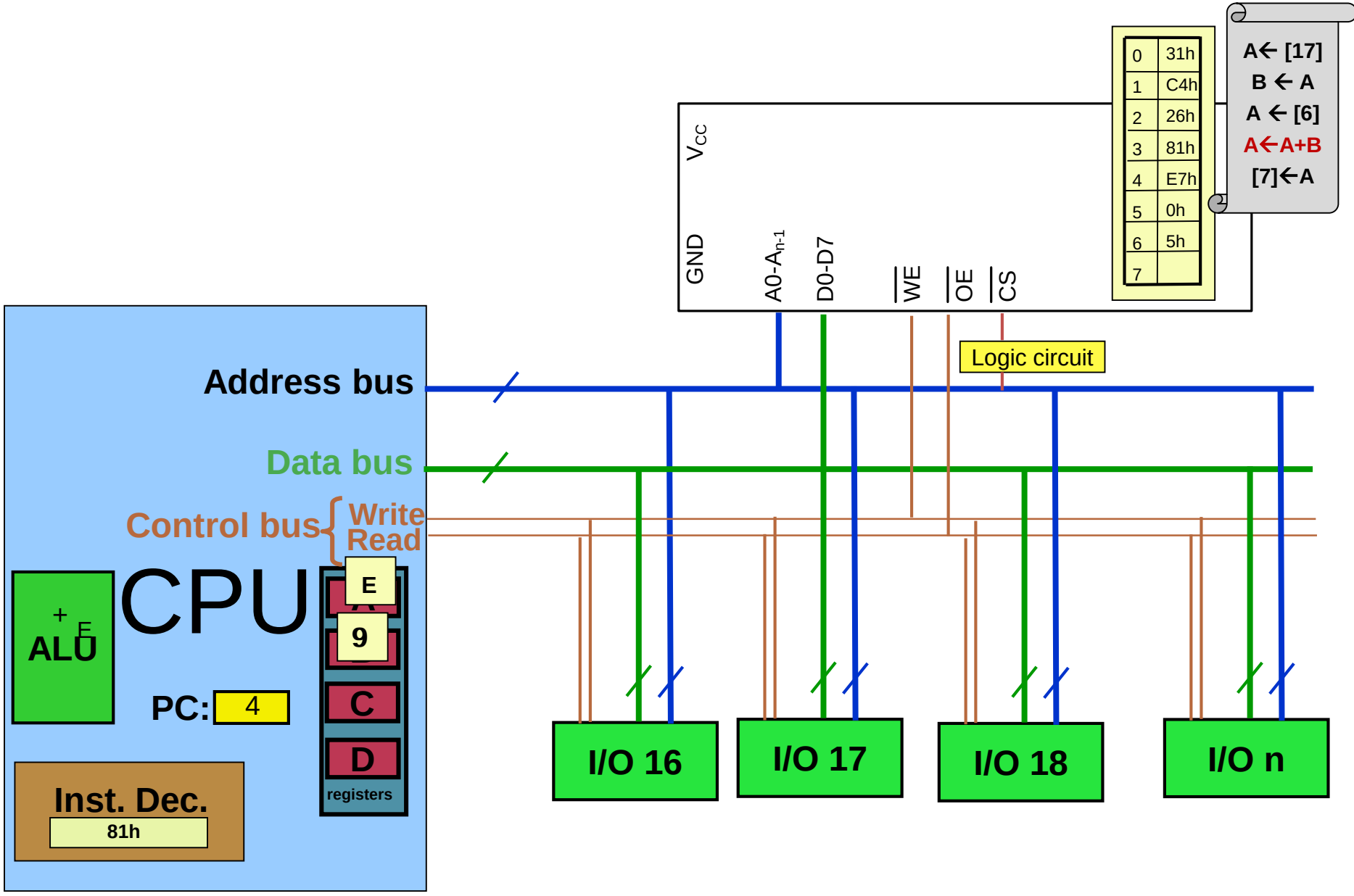
Instruction Fetch➔(Decode)➔Execute in CPU



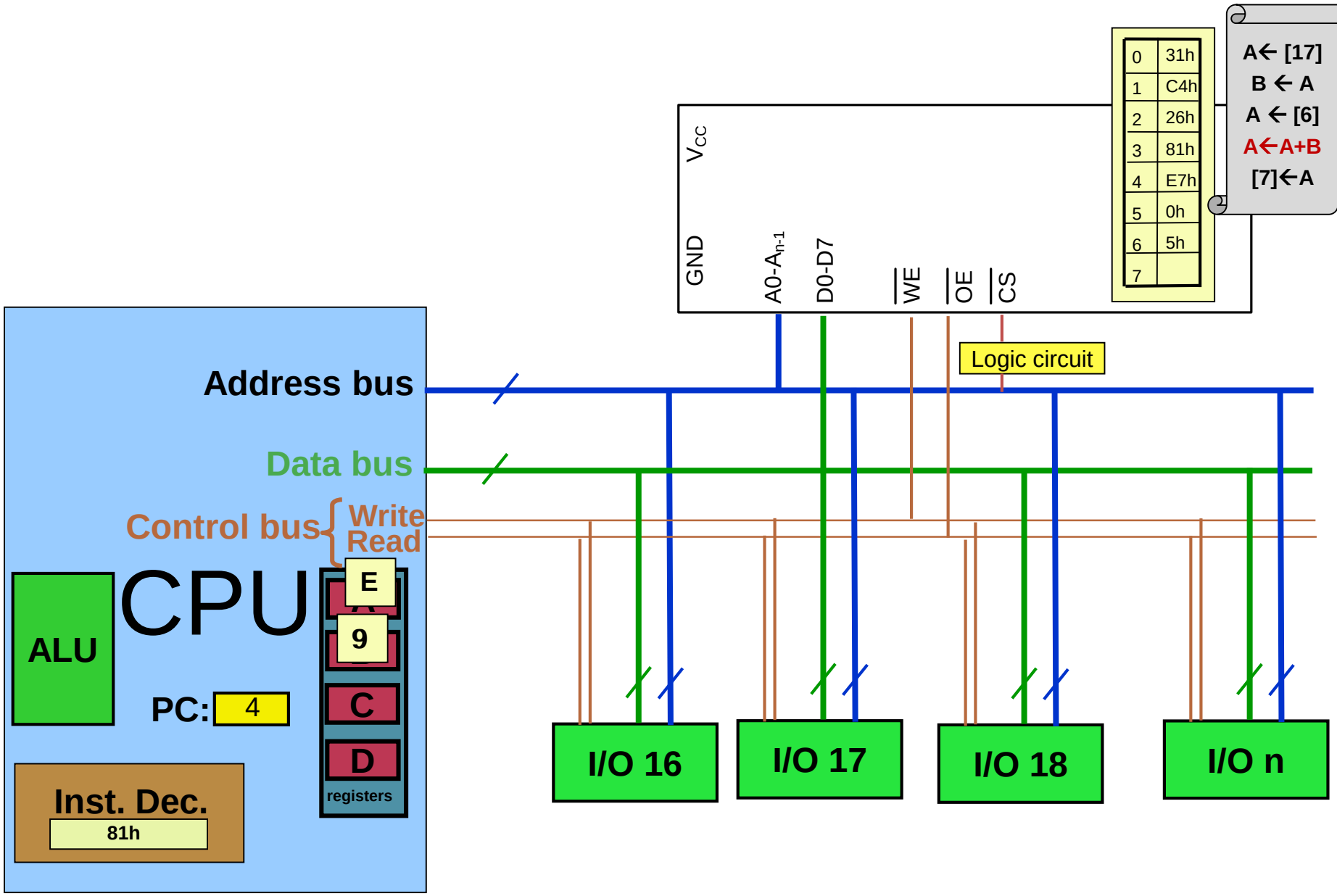
Instruction Fetch➔(Decode)➔Execute in CPU



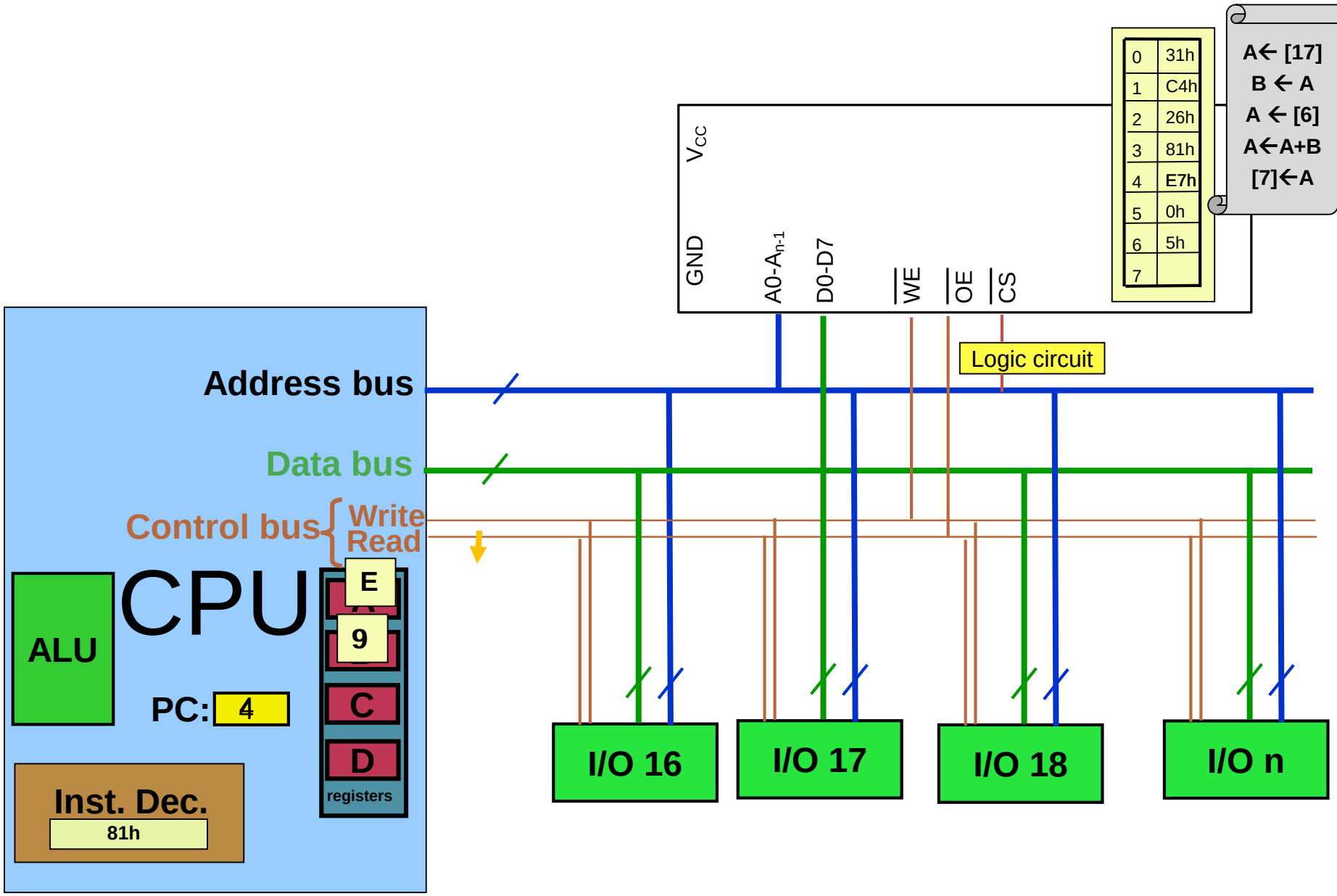
Instruction Fetch➔(Decode)➔Execute in CPU



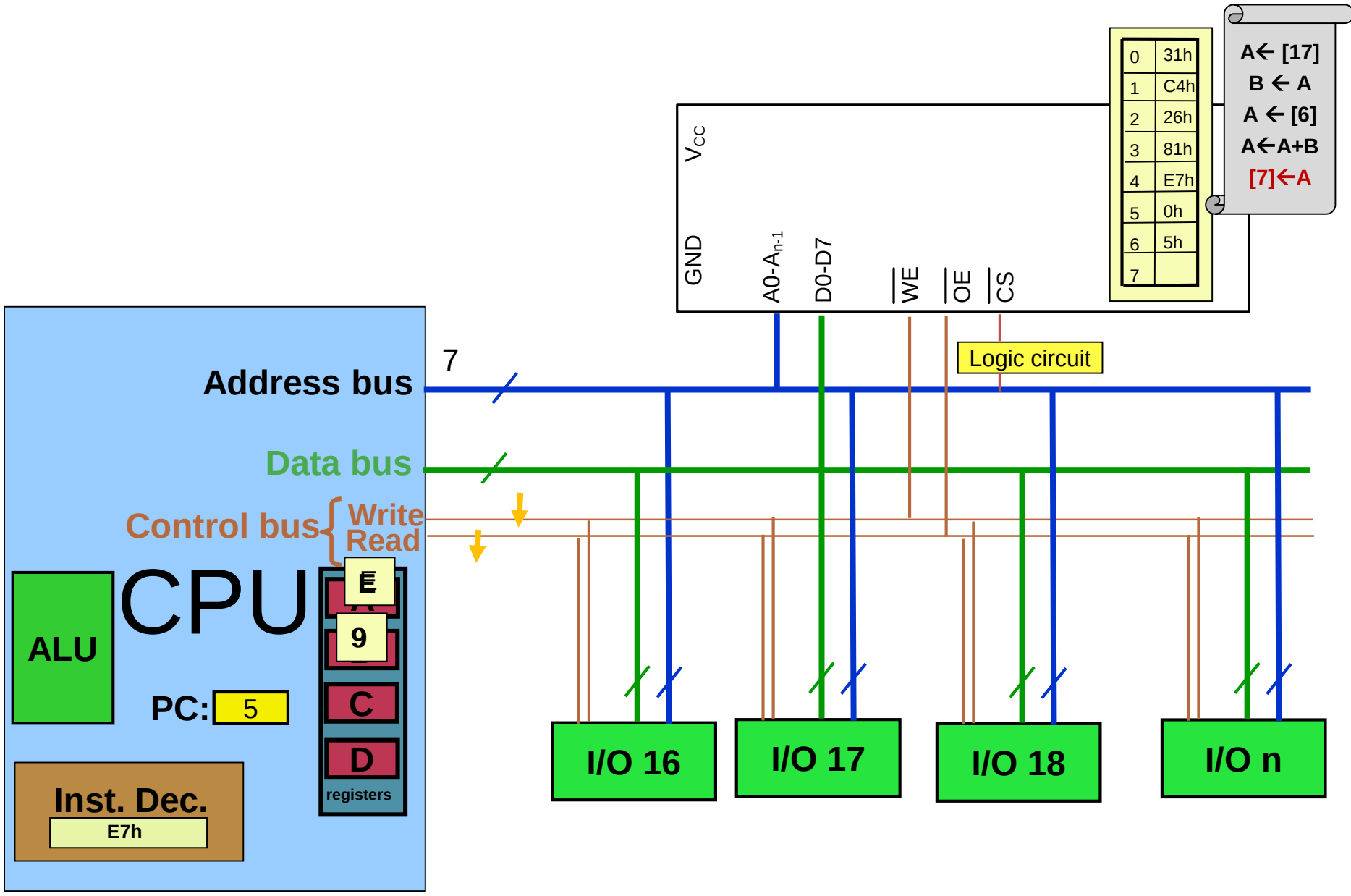
Instruction Fetch➔(Decode)➔Execute in CPU



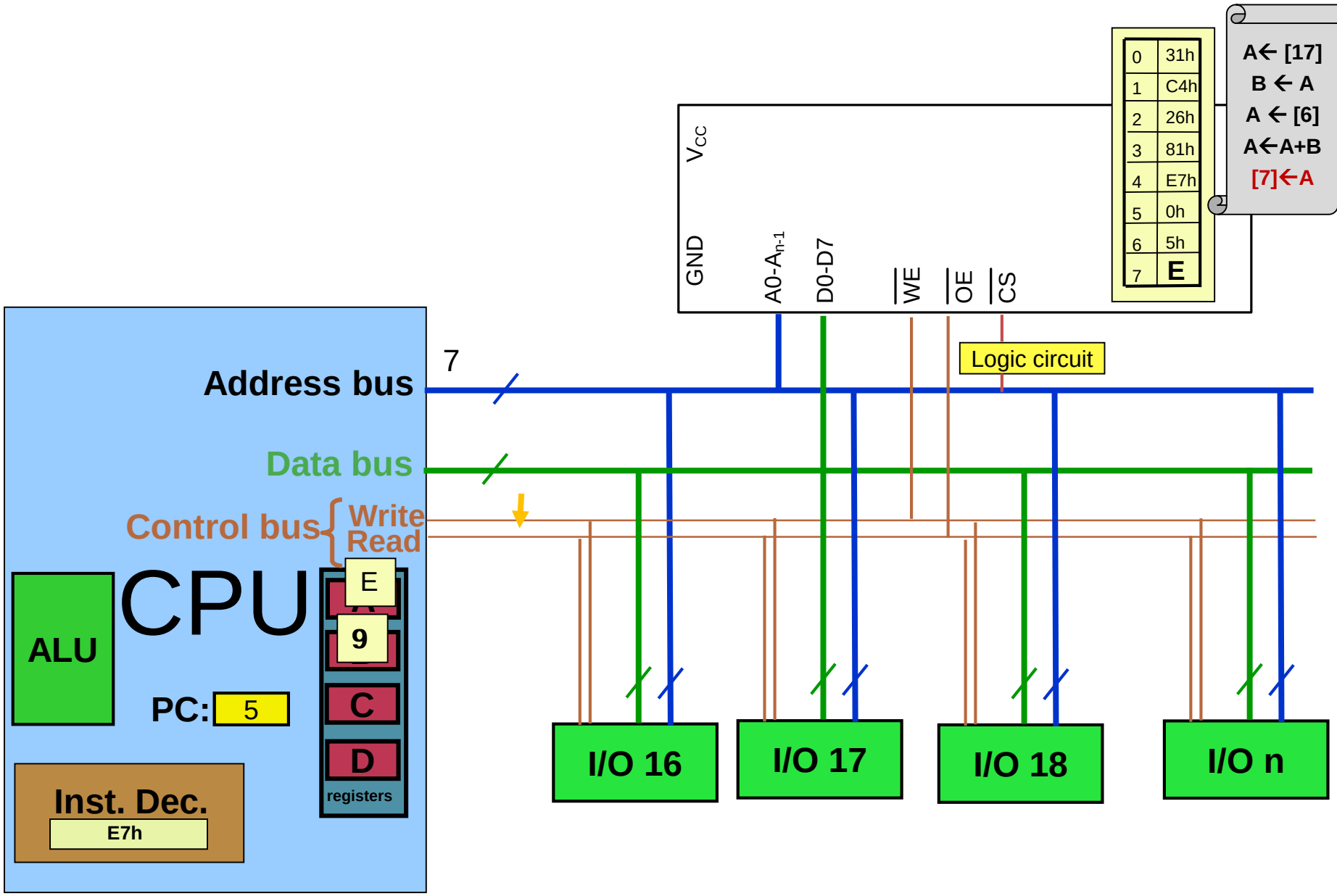
Instruction Fetch➔(Decode)➔Execute in CPU



Instruction Fetch➔(Decode)➔Execute in CPU

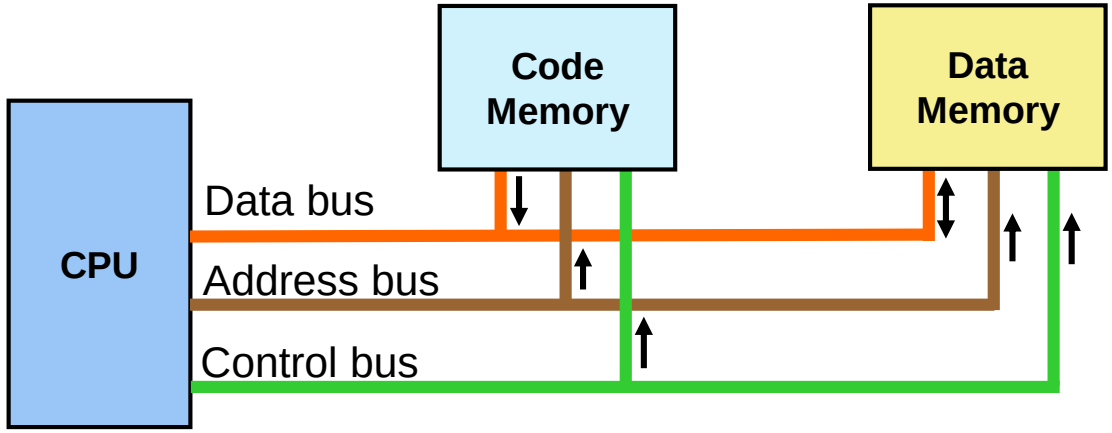


Instruction Fetch➔(Decode)➔Execute in CPU

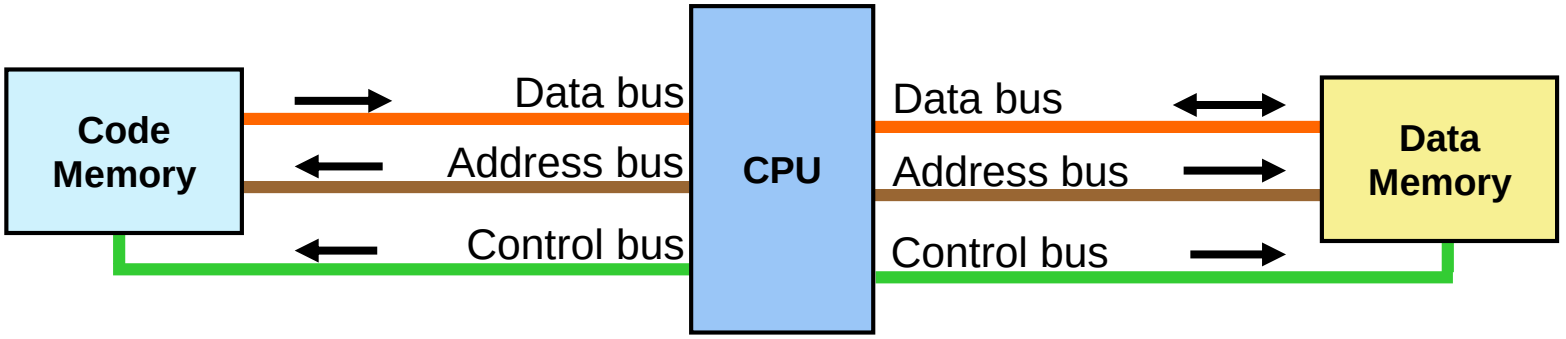


von Neumann vs. Harvard Architecture

- von Neumann architecture



- Harvard architecture



Embedded Systems

- What is an Embedded System?
  - Application-specific computer system
  - Built into a larger system
- Why add a computer to the larger system?
  - Better performance
  - More functions and features
  - Lower cost
  - More dependability
- Economics
  - Microcontrollers (used for embedded computers) are high-volume, so recurring cost is low
  - Nonrecurring cost dominated by software development
- Networks
  - Often embedded system will use multiple processors communicating across a network to lower parts and assembly costs and improve reliability

Options for Building Embedded Systems

Dedicated Hardware

Software Running on Generic Hardware

Implementation	Design Cost	Unit Cost	Upgrades & Bug Fixes	Size	Weight	Power	System Speed
Discrete Logic	low	mid	hard	large	high	?	very fast
ASIC	high ($500K/ mask set)	very low	hard	tiny - 1 die	very low	low	extremely fast
Programmable logic – FPGA, PLD	low	mid	easy	small	low	medium to high	very fast
Microprocessor + memory + peripherals	low to mid	mid	easy	small to med.	low to moderate	medium	moderate
Microcontroller (int. memory & peripherals)	low	mid to low	easy	small	low	medium	slow to moderate
Embedded PC	low	high	easy	medium	moderate to high	medium to high	fast

Example Embedded System: Bike Computer

- Functions
  - Speed and distance measurement
- Constraints
  - Size
  - Cost
  - Power and Energy
  - Weight
- Inputs
  - Wheel rotation indicator
  - Mode key
- Output
  - Liquid Crystal Display
- Low performance MCU
  - 8-bit, 10 MIPS

Gasoline Automobile Engine Control Unit

- Functions
  - Fuel injection
  - Air intake setting
  - Spark timing
  - Exhaust gas circulation
  - Electronic throttle control
  - Knock control
- Constraints
  - Reliability in harsh environment
  - Cost
  - Weight

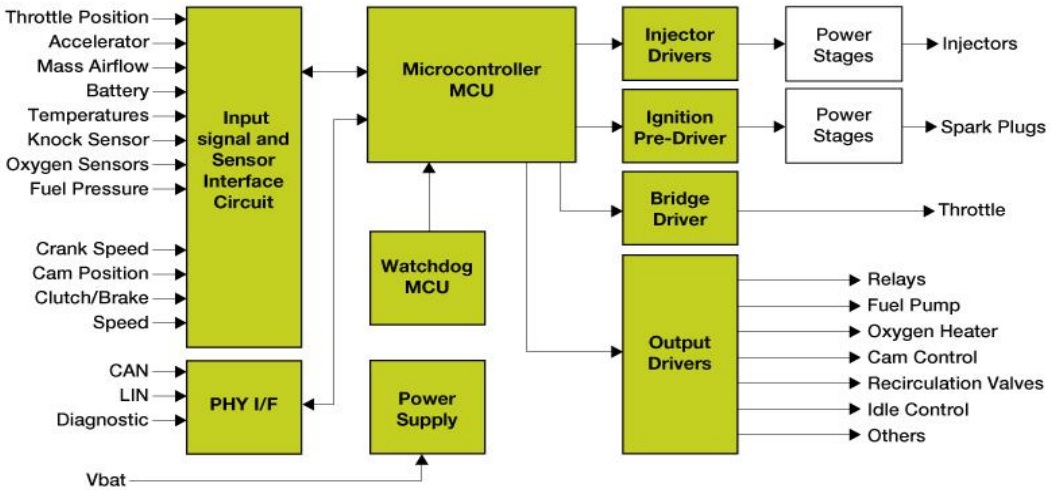
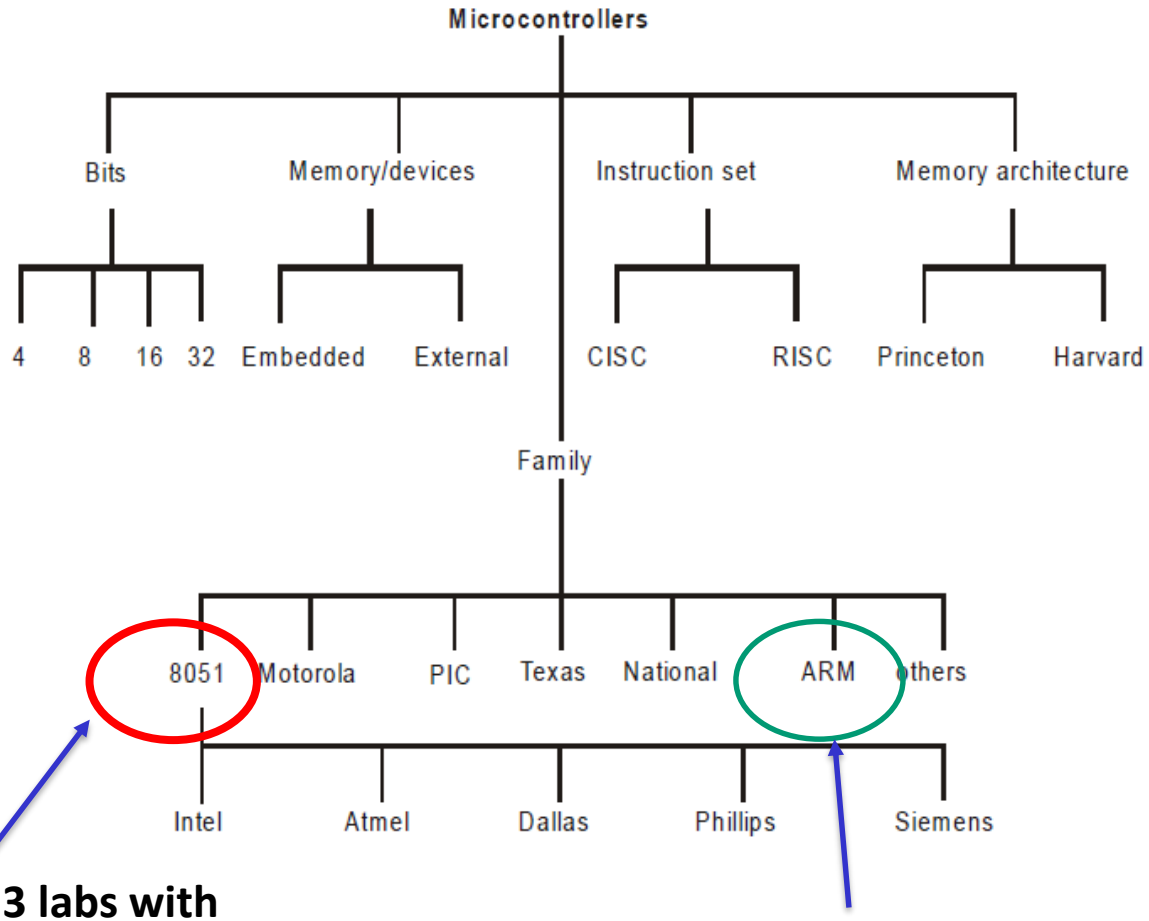


Image courtesy of Freescale

- Many Inputs and Outputs
  - Discrete sensors & actuators
  - Network interface to rest of car
- High Performance MCU
  - 32-bit, 3 MB flash memory, 150 - 300 MHz

Microcontoller Families



Part I: 3 labs with Assembly Programming and Proteus Simulation

Part II: 3 Labs + Project with C programming

Intel 8051

- 1981, Intel MCS-51
- The 8051 became popular after Intel allowed other manufacturers to make and market a flavor of the 8051.
  - different speed, amount of on-chip ROM
  - code-compatible with the original 8051
  - form a 8051 family

8051 Features

Feature	Quantity	Notes
ROM	4K bytes	a fixed program
RAM	128 bytes	temporary data
Timers	2	Timer/counter 0,1
I/O pins	32	P0,P1,P2,P3
Serial port	1	TxD, RxD
Interrupt sources	6	

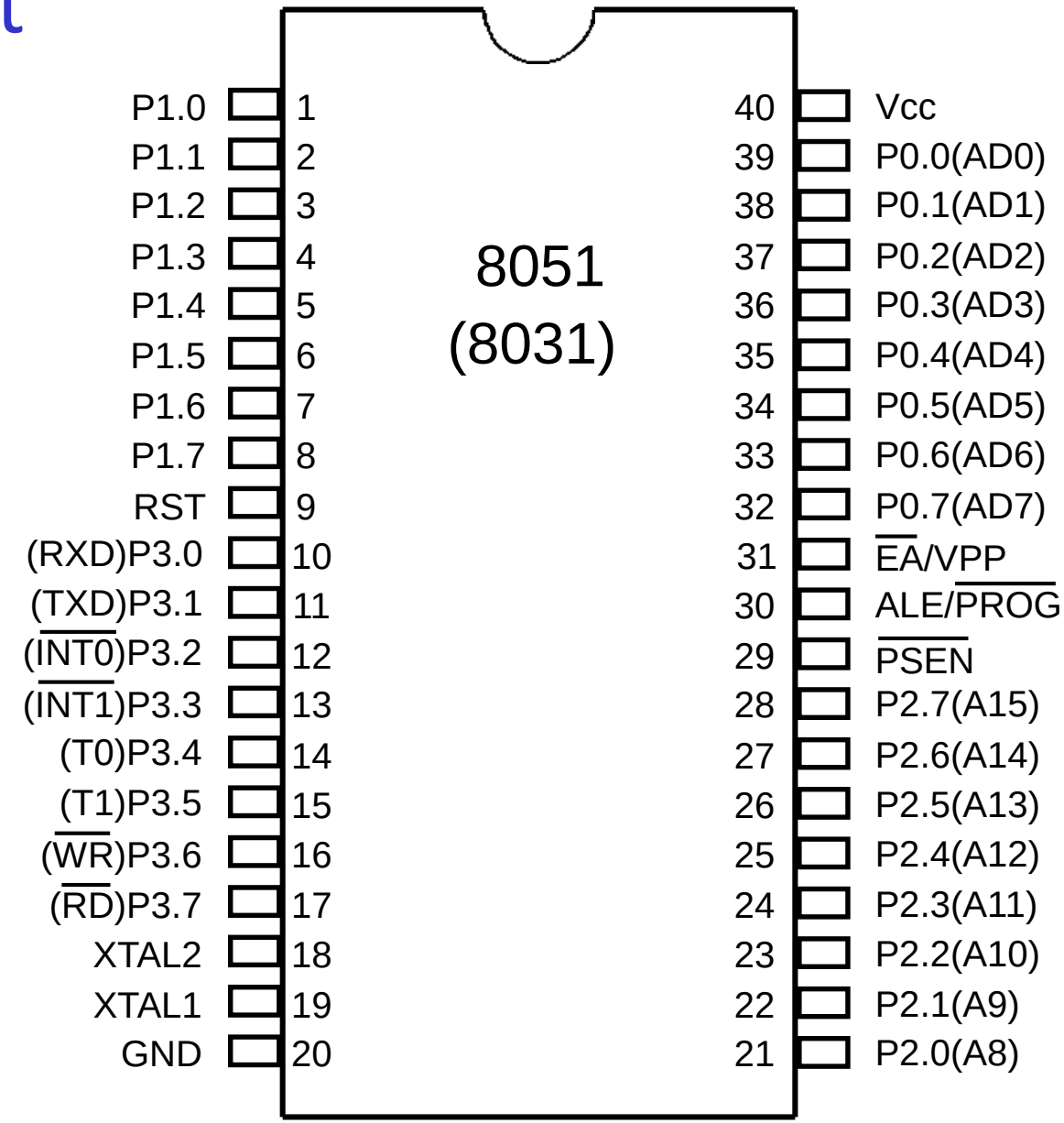
Comparison of 8051 Members

Table 1-4 Comparison of the 8051 Family Members

Feature	**8051**	**8052**	**8031**
ROM (program space in bytes)	4K	8K	0
RAM (bytes)	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

- 8031 is referred as ROM-less 8051
- To use ROM you must add external ROM to it but you lose two ports

8051 Layout


8051
(8031)
P1.0 1
P1.1 2
P1.2 3
P1.3 4
P1.4 5
P1.5 6
P1.6 7
P1.7 8
RST 9
(RXD)P3.0 10
(TXD)P3.1 11
(INT0)P3.2 12
(INT1)P3.3 13
(T0)P3.4 14
(T1)P3.5 15
(WR)P3.6 16
(RD)P3.7 17
XTAL2 18
XTAL1 19
GND 20
40 Vcc
39 P0.0(AD0)
38 P0.1(AD1)
37 P0.2(AD2)
36 P0.3(AD3)
35 P0.4(AD4)
34 P0.5(AD5)
33 P0.6(AD6)
32 P0.7(AD7)
31 EA/VPP
30 ALE/PROG
29 PSEN
28 P2.7(A15)
27 P2.6(A14)
26 P2.5(A13)
25 P2.4(A12)
24 P2.3(A11)
23 P2.2(A10)
22 P2.1(A9)
21 P2.0(A8)

8051 Block Diagram

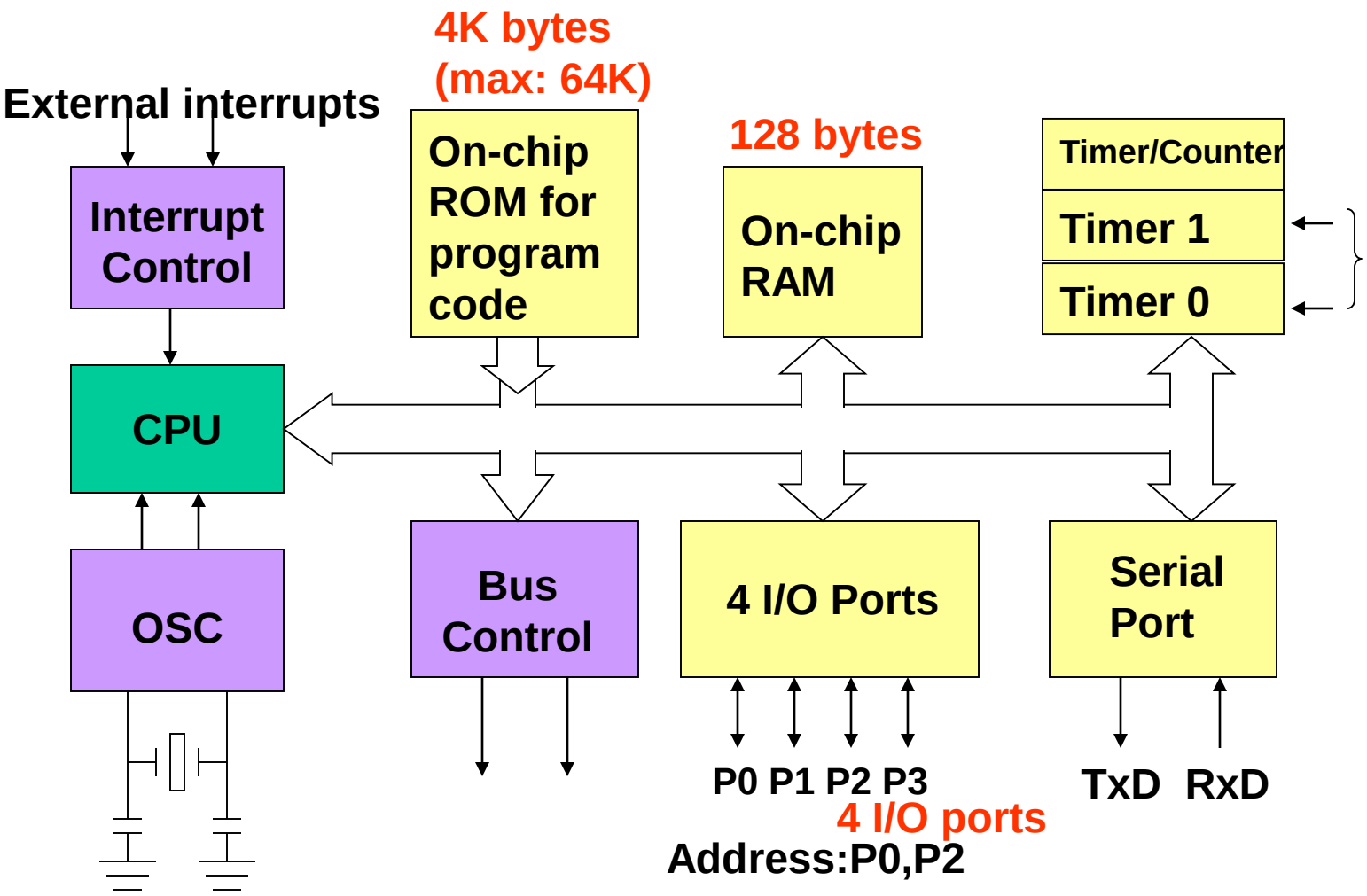


Figure 1-2. Inside the 8051 Microcontroller Block Diagram

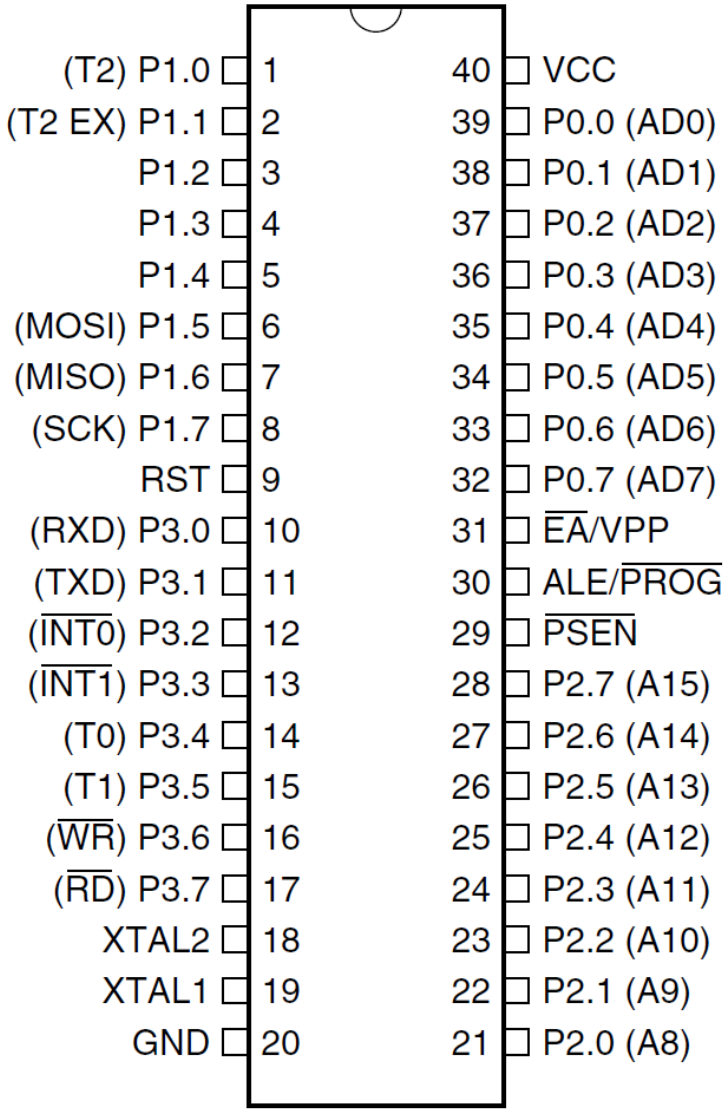
Different 8051 Products

- Distinguished by types of ROM:
- 8751 microcontroller — 4k bytes UV-EPROM, PROM burner, UV-EPROM eraser
- AT89C51 from Atmel Corporation — flash memory, PROM burner only
- DS89C4x0 from Dallas Semiconductor — flash memory, r/w from/to COM port
- DS5000 from Dallas Semiconductor — NV-RAM, r/w from/to PC serial port
- OTP (one-time-programmable) version of the 8051 — for large market
- 8051 family from Philips

8051 Products

- The long proven 80C51 continues to add variants, and suppliers. Even Microchip now sells 80C51
- A couple of vendors now deliver Dual Core 8051's
- With expanding QuadSPI memory choices, we also see the first C51's offering Execute-In-Place from 8 pin SPI memory
- 12 Bit ADC is moving to be more mainstream. Advanced Analog performance (16/20+ ADC bits) is available from Analog Devices, TI, SiLabs, Maxim, Nuvoton etc.
- Small packages: 2mm(+)/3mm/4mm etc, and SO8/MSOP10/SO14/SO16 are now design choices.
- Another trend sees C51's moving into system chips, as the on-chip controller. USB 2.0 480MBaud, USB-OTG, Ethernet, Power Line Comms, and BlueTooth systems are all seeing C51 cores.
- Silabs and NXP drive the 80C51 in FLASH well under $1 and to 14/11/8pins
- Single cycle 80C51 cores are expanding all the time.
- On chip regulators, and low power, Calibrated Oscillators are also emerging trends.
- Many far-east chip vendors and foundries are now backing the 80C51.

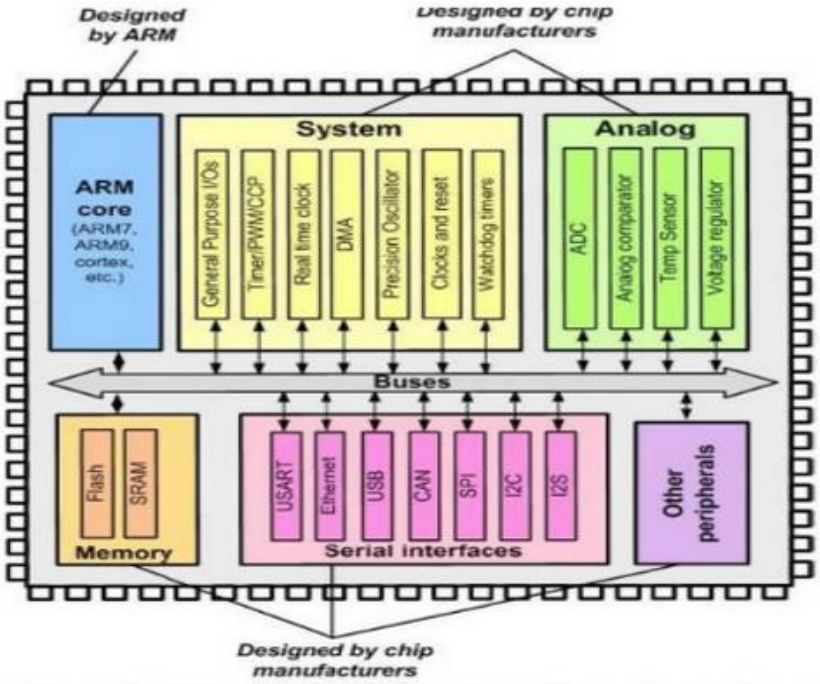
Atmel 8051 Family – Atmel AT89S52

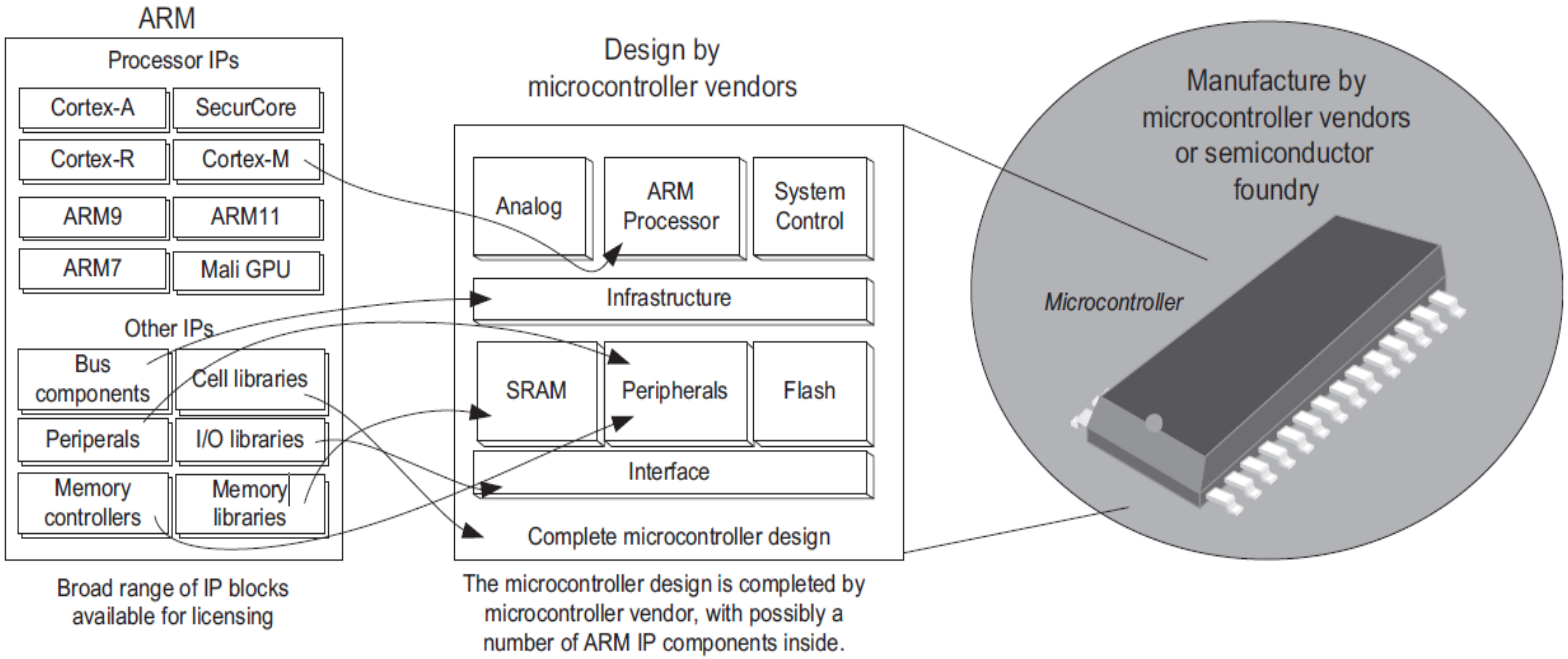


- Compatible with MCS®-51 Products
- 8K Bytes of In-System Programmable (ISP) Flash Memory
  - Endurance: 10,000 Write/Erase Cycles
- 4.0V to 5.5V Operating Range
- Fully Static Operation: 0 Hz to 33 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Full Duplex UART Serial Channel
- Low-power Idle and Power-down Modes
- Interrupt Recovery from Power-down Mode
- Watchdog Timer
- Dual Data Pointer
- Power-off Flag
- Fast Programming Time
- Flexible ISP Programming (Byte and Page Mode)
- Green (Pb/Halide-free) Packaging Option

ARM

- ARM is one of the most licensed and thus widespread processor cores in the world
- Used especially in portable devices due to low power consumption and reasonable performance (MIPS / watt)


Designed by ARM
Designed by chip manufacturers
ARM core (ARM7, ARM9, cortex, etc.)
System
General Purpose I/Os
Timer/PWM/CCP
Real time clock
DMA
Precision Oscillator
Clocks and reset
Watchdog timers
Analog
ADC
Analog comparator
Temp Sensor
Voltage regulator
Buses
Flash
SRAM
Memory
USART
Ethernet
USB
CAN
SPI
I2C
I2S
Serial interfaces
Other peripherals
Designed by chip manufacturers


ARM
Processor IPs
Cortex-A
SecurCore
Cortex-R
Cortex-M
ARM9
ARM11
ARM7
Mali GPU
Other IPs
Bus components
Cell libraries
Peripherals
I/O libraries
Memory controllers
Memory libraries
Broad range of IP blocks available for licensing
Design by microcontroller vendors
Analog
ARM Processor
System Control
Infrastructure
SRAM
Peripherals
Flash
Interface
Complete microcontroller design
The microcontroller design is completed by microcontroller vendor, with possibly a number of ARM IP components inside.
Manufacture by microcontroller vendors or semiconductor foundry
Microcontroller

ARM History

Year	Billion units	Relative size
2015	15	
2014	12	
2013	10	
2012	8.7	
2011	7.9	
2010	6.1	
2009	3.9	
2008	4.0	
2007	2.9	
2006	2.4	
2005	1.662	
2004	1.272	
2003	0.782	
2002	0.456	
2001	0.420	
2000	0.367	
1999	0.175	
1998	0.051	
1997	0.009	
Total	**78.094**	

- Acorn Computers
  - British computer company based in Cambridge.
  - Set-up in 1978 by Chris Curry and Hermann Hauser.
  - Most famous computer was the BBC Micro (1980)
  - First RISC machine – 1985
  - First RISC based home computer – Archimedes - 1987
- ARM
  - Founded in November 1990 by Acorn Comp, Apple, VLSI Tech.
  - Licencing – 1992
  - ARM holdings float on to Nasdaq and London Stock Exch. – 1998
  - 1 billion ARM based chips have been shipped – 2002
  - Energy efficient Cortex chips have been announced – 2004
  - 10 billion chips shipped (**95%** of smartphones, **35%** of digital televisions and set-top boxes and **10%** of mobile computers) – 2009
  - 64-bit capable ARMv8 architecture is introduced – 2011
  - ARM named as fifth most innovative company in the world – 2014
  - ARM reported profit of $448.4 million – 2015
  - SoftBank acquired ARM for $31 billion - 2016

Sample ARM powered products

iPhone 5

The A6 processor is the first Apple System-on-Chip (SoC) to use a custom design, based off the **ARMv7** instruction set.

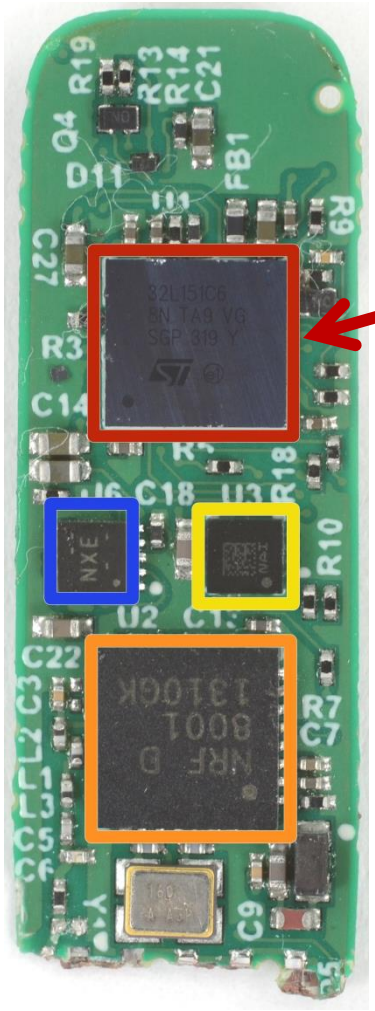
iPhone 6

The A8 processor is the first 64-bit ARM based SoC. It supports **ARM A64, A32, and T32** instruction set.

Apple Watch

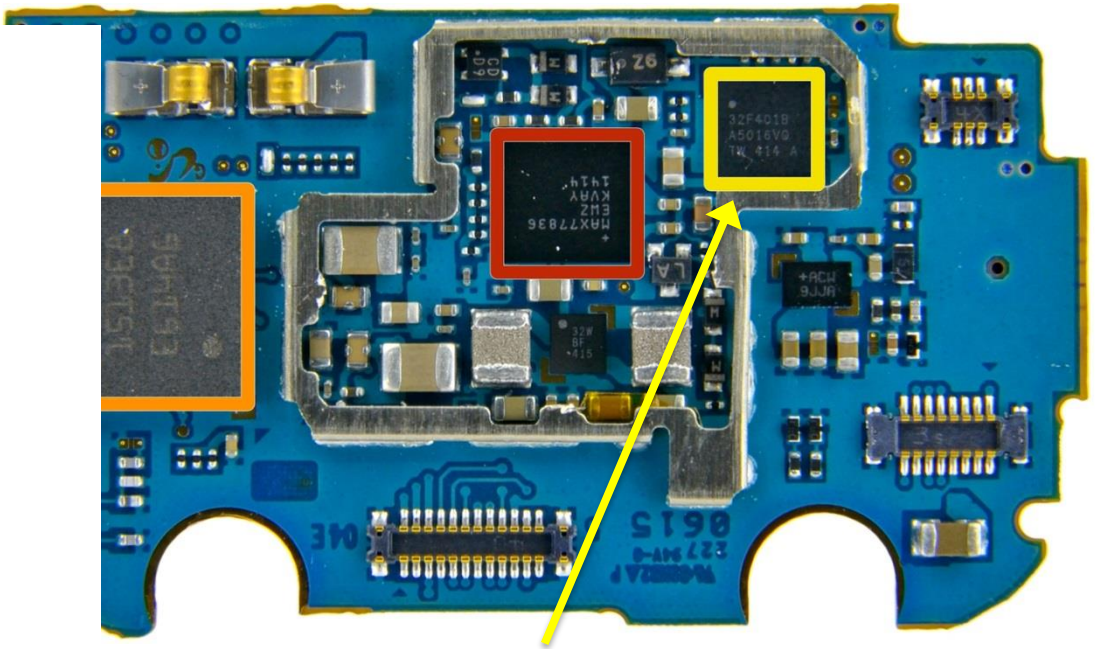
- Apple S1 Processor
  - **32-bit ARMv7-A** compatible
  - # of Cores: **1**
  - CMOS Technology: 28 nm
  - L1 cache 32 KB data
  - L2 cache 256 KB
  - GPU PowerVR SGX543

Fitbit Flex Teardown



STMicroelectronics 32L151C6 Ultra Low Power ARM Cortex M3 Microcontroller

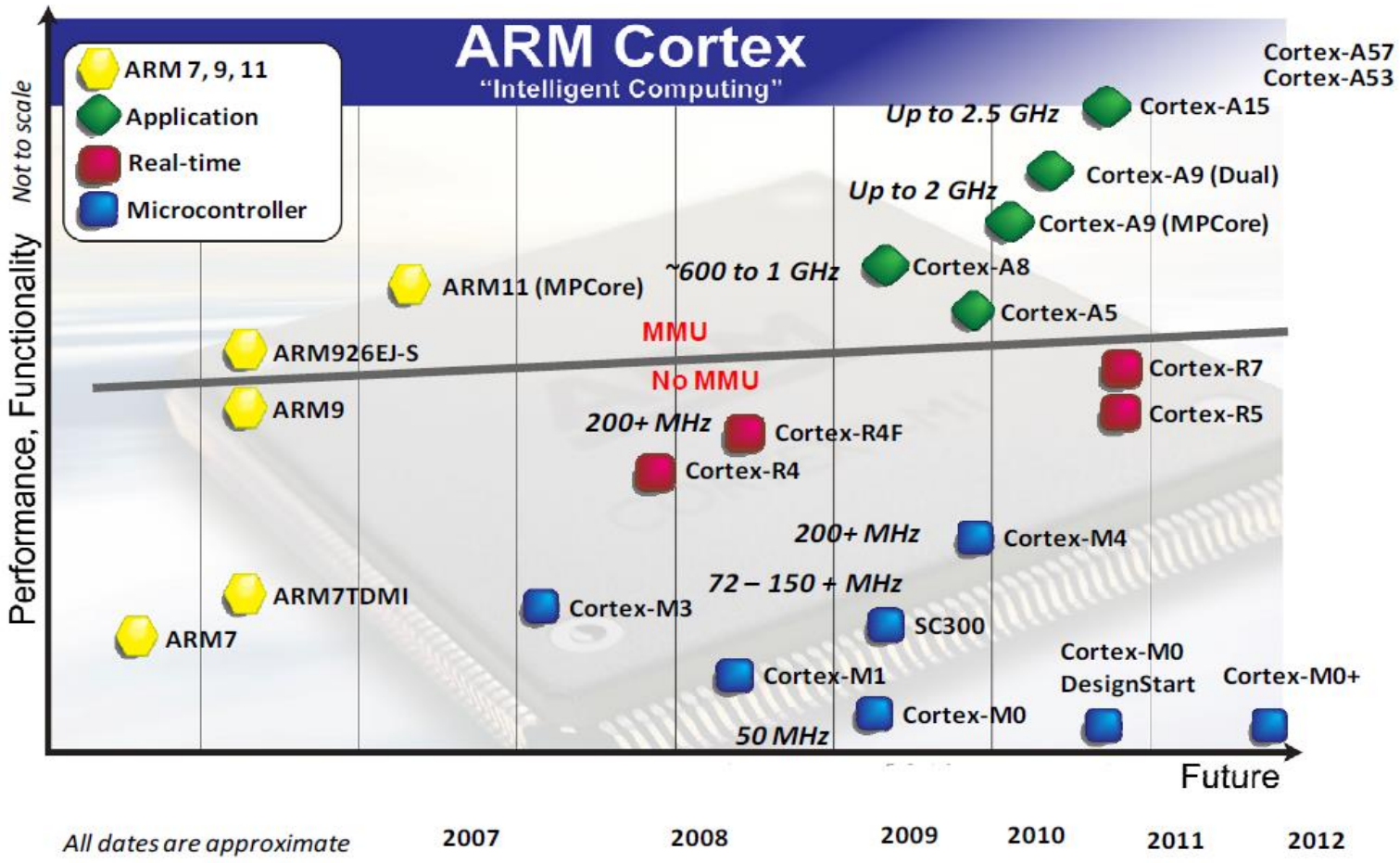
Samsung Galaxy Gear



- STMicroelectronics STM32F401B **ARM-Cortex M4** MCU with 128KB Flash

Family	Architecture	Cores
ARM7TDMI	ARMv4T	ARM7TDMI(S)
ARM9 ARM9E	ARMv5TE(J)	ARM926EJ-S, ARM966E-S
ARM11	ARMv6 (T2)	ARM1136(F), 1156T2(F)-S, 1176JZ(F), ARM11 MPCore™
Cortex-A	ARMv7-A	Cortex-A5, A7, A8, A9, A15
Cortex-R	ARMv7-R	Cortex-R4(F)
Cortex-M	ARMv7-M	Cortex-M3, M4
	ARMv6-M	Cortex-M1, M0
NEW !	ARMv8-A	64 Bit

Features	ARM7TDMI-S	Cortex-M3
Architecture	ARMv4T (von Neumann)	ARMv7-M (Harvard)
ISA Support	Thumb / ARM	Thumb / Thumb-2
Pipeline	3-Stage	3-Stage + branch speculation
Interrupts	FIQ / IRQ	NMI + 1 to 240 Physical Interrupts
Interrupt Latency	24-42 Cycles	12 Cycles
Sleep Modes	None	Integrated
Memory Protection	None	8 region Memory Protection Unit
Dhrystone	0.95 DMIPS/MHz (ARM mode)	1.25 DMIPS/MHz
Power Consumption	0.28mW/MHz	0.19mW/MHz
Area	0.62mm2 (Core Only)	0.86mm2 (Core & Peripherals)*


ARM Cortex
"Intelligent Computing"
ARM 7, 9, 11
Application
Real-time
Microcontroller
Not to scale
Performance, Functionality
Cortex-A57
Cortex-A53
Up to 2.5 GHz
Cortex-A15
Cortex-A9 (Dual)
Up to 2 GHz
Cortex-A9 (MPCore)
~600 to 1 GHz
Cortex-A8
Cortex-A5
ARM11 (MPCore)
MMU
ARM926EJ-S
No MMU
Cortex-R7
Cortex-R5
ARM9
200+ MHz
Cortex-R4F
Cortex-R4
200+ MHz
Cortex-M4
72 – 150 + MHz
ARM7TDMI
Cortex-M3
SC300
ARM7
Cortex-M0
DesignStart
Cortex-M0+
Cortex-M1
Cortex-M0
50 MHz
Future
All dates are approximate
2007
2008
2009
2010
2011
2012

Where to use?

Cortex M-series: Cost-sensitive solutions for microcontroller applications, system clock<200MHz

Cortex M0 – Ultra low-power, oltra low gate count,

Cortex M1 – First ARM processor designed specifically for implementation in FPGAs.

Cortex M3 – High performance and energy efficiency. Microcontroller applications.

Cortex M4 – Embedded processor for DSP

Cortex M7 – 2x Performance of M4

Cortex R-series: Exceptional performance for real-time applications, system clock<600MHz

Cortex R4 - First embedded real-time processor based on the ARMv7-R architecture for high-volume deeply-embedded System-on-Chip applications such as hard disk, wireless baseband processors, electronic control units for automotive systems.

Cortex R5 – Extends the feature set of Cortex-R4, incereased efficiency and reliability.

Cortex R7 – High-performance dual core

Cortex A-series: High performance processors for open Operating sys, system clock>1GHz

Cortex A5 – Power and cost sensitive applications, smartphones.

Cortex A8 – Suitable for high-end phones, printers, DTVs

Cortex A9 – 1 to 4 cores. High performance-low power devices

Cortex A15 – Ultra low-power. Suitable for mobile computing, wireless infrastructure

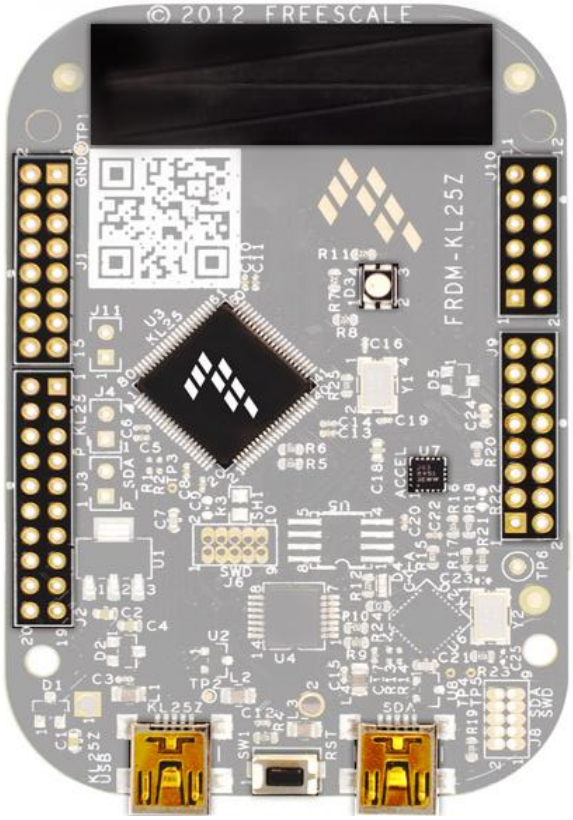
ARM 7 – Introduced in 1994, used for simple 32-bit devices.

ARM9 – Was most popular ARM family (over 5 billion sold), used for smarthphones, HDD controllers, set top box, etc.

ARM11 - Extreme low power, many of today's smartphones.

Target Board - FRDM-KL25Z

- 32-bit Cortex M0+ Processor Core
- Freescale Kinetis MKL25Z128VLK4 processor
  - Extremely low power use
  - 48 MHz max clock
  - On-chip 128 KB ROM, 16 KB RAM
  - Wide range of peripherals, including USB on-the-go
- FRDM-KL25Z board
  - $13 (USD)
  - Peripherals: 3-axis accelerometer, RGB LED, capacitive touch slider
  - Expansion ports are compatible with Arduino shield ecosystem – endless opportunities, low-cost hardware



Microcontroller Vendors

Leading MCU Suppliers ($M)

2016 Rank	Company	2015	2016	% Change	% Marketshare
1	**NXP***	1,350	2,914	116%	19%
2	**Renesas**	2,560	2,458	-4%	16%
3	**Microchip****	1,355	2,027	50%	14%
4	**Samsung**	2,170	1,866	-14%	12%
5	**ST**	1,514	1,573	4%	10%
6	**Infineon**	1,060	1,106	4%	7%
7	**Texas Instruments**	820	835	2%	6%
8	**Cypress*****	540	622	15%	4%

*Acquired Freescale in December 2015.

**Purchased Atmel in April 2016.

***Includes full year of sales from Spansion acquisition in March 2015.

Source: IC Insights, company reports

- 8-bit Atmel
- 8-bit 8051
- 8-bit PIC
- 16-bit PIC
- 32-bit PIC
- 32-bit ARM Cortex-M

Programming Languages for Embbedded Systems

- https://www.qt.io/embedded-development-talk/embedded-software-programming-languages-pros-cons-and-comparisons-of-popular-languages
- C Programming
  - **The basics:** Developed in the early 1970s, C is a compiled language that serves as a building block of many other languages.
  - **Pros**: C is an efficient and widely used programming language. Industry estimates say 80% of embedded systems use the C programming language.
  - **Cons**: Requires developers to understand and use technical coding techniques that can be complicated.