Maynooth University
National University of Ireland Maynooth

CS433 Modern Architectures

Video 2

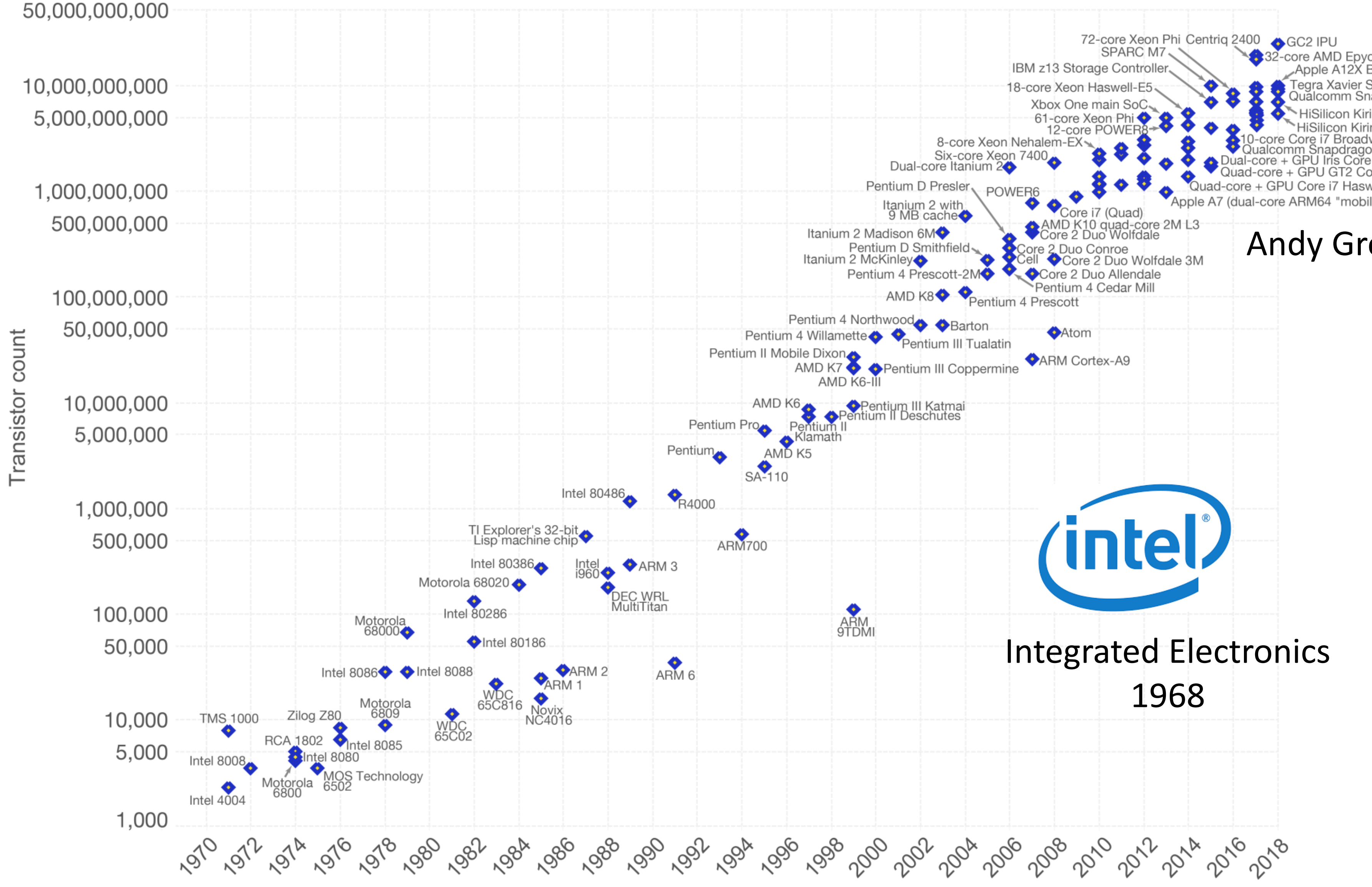
The machine cycle

Topic 1.1: Moore’s Law

“The number of transistors that can be put on a given area of Silicon doubles (roughly) every two years”

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

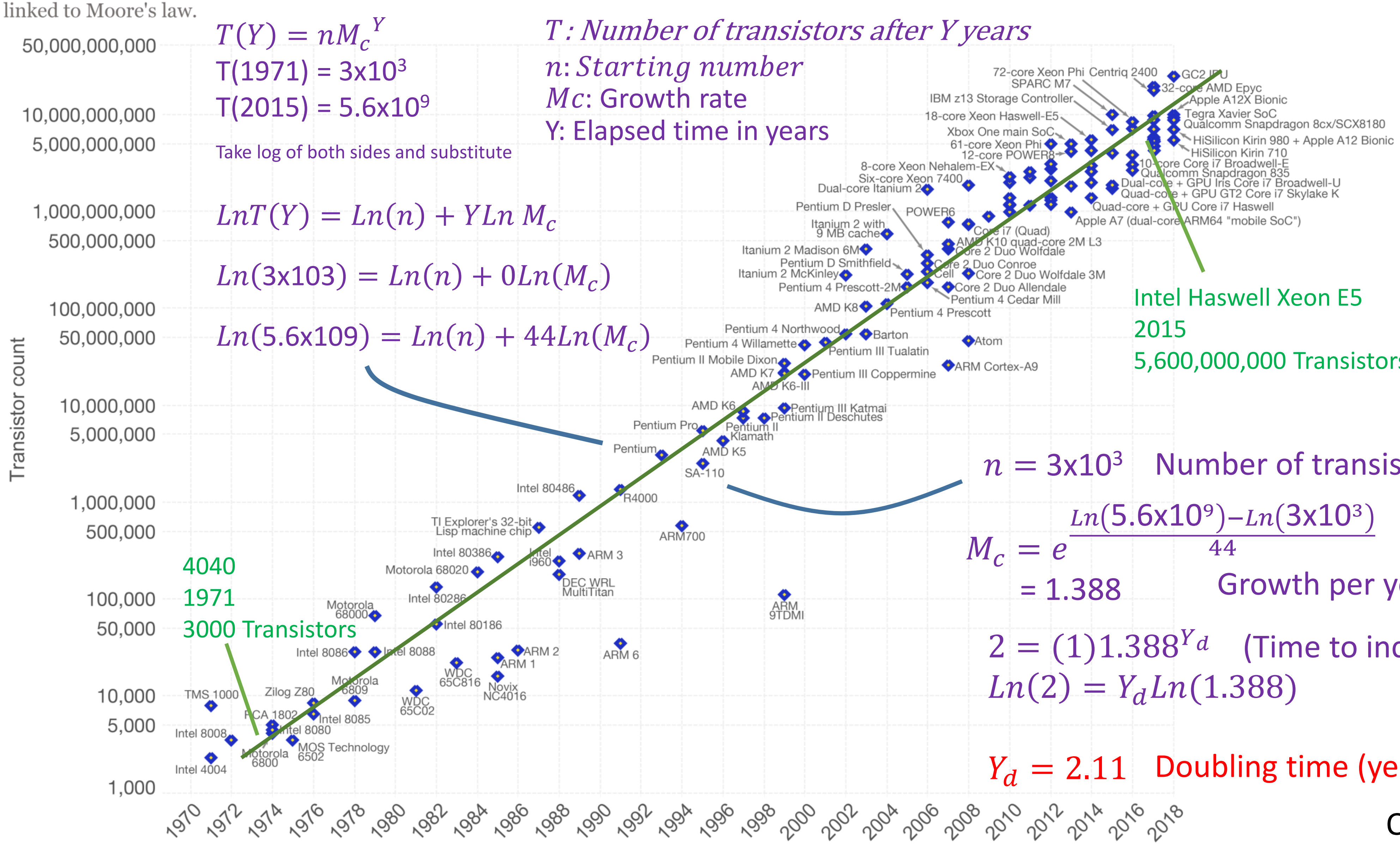
Integrated Electronics
1968

Andy Grove, Robert Noyce and Gordon Moore

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



$T(Y) = nM_c{}^{Y}$

T(1971) = 3x10^3

T(2015) = 5.6x10^9

Take log of both sides and substitute

T : Number of transistors after Y years

n: Starting number

Mc: Growth rate

Y: Elapsed time in years

$LnT(Y) = Ln(n) + YLn\ M_c$

$Ln(3x103) = Ln(n) + 0Ln(M_c)$

$Ln(5.6x109) = Ln(n) + 44Ln(M_c)$

Intel Haswell Xeon E5
2015
5,600,000,000 Transistors

4040
1971
3000 Transistors

n = 3x10^3 Number of transistors at start

$$M_c = e^{\frac{Ln(5.6x10^9) - Ln(3x10^3)}{44}}$$

= 1.388 Growth per year

$2 = (1)1.388^{Y_d}$ (Time to increase x2)

$Ln(2) = Y_d Ln(1.388)$

Y_d = 2.11 Doubling time (years)

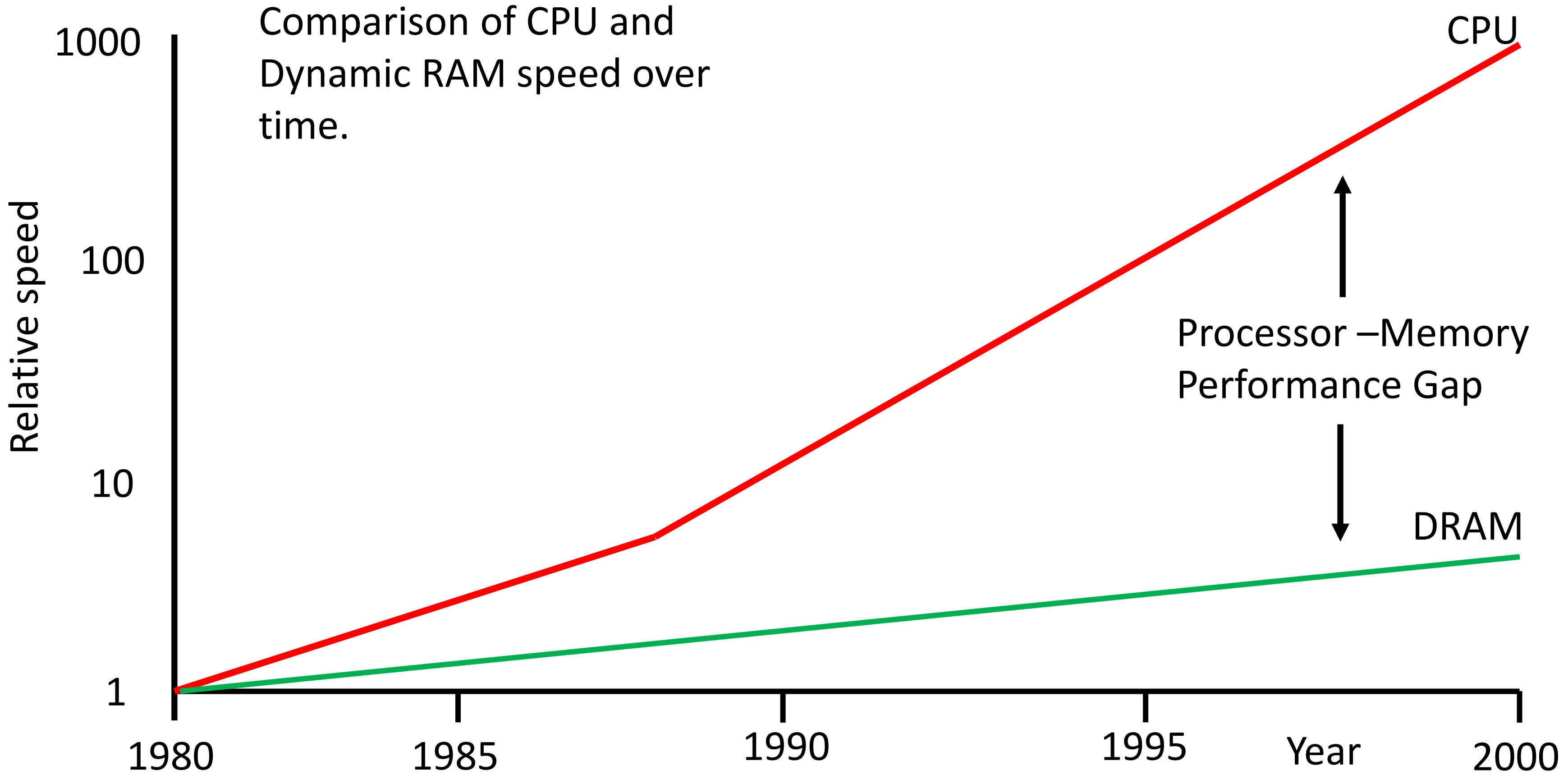
You should check and validate information presented by experiment and analysis. This will be a common theme of the module.

Similar calculations for "Covid 19 R value"

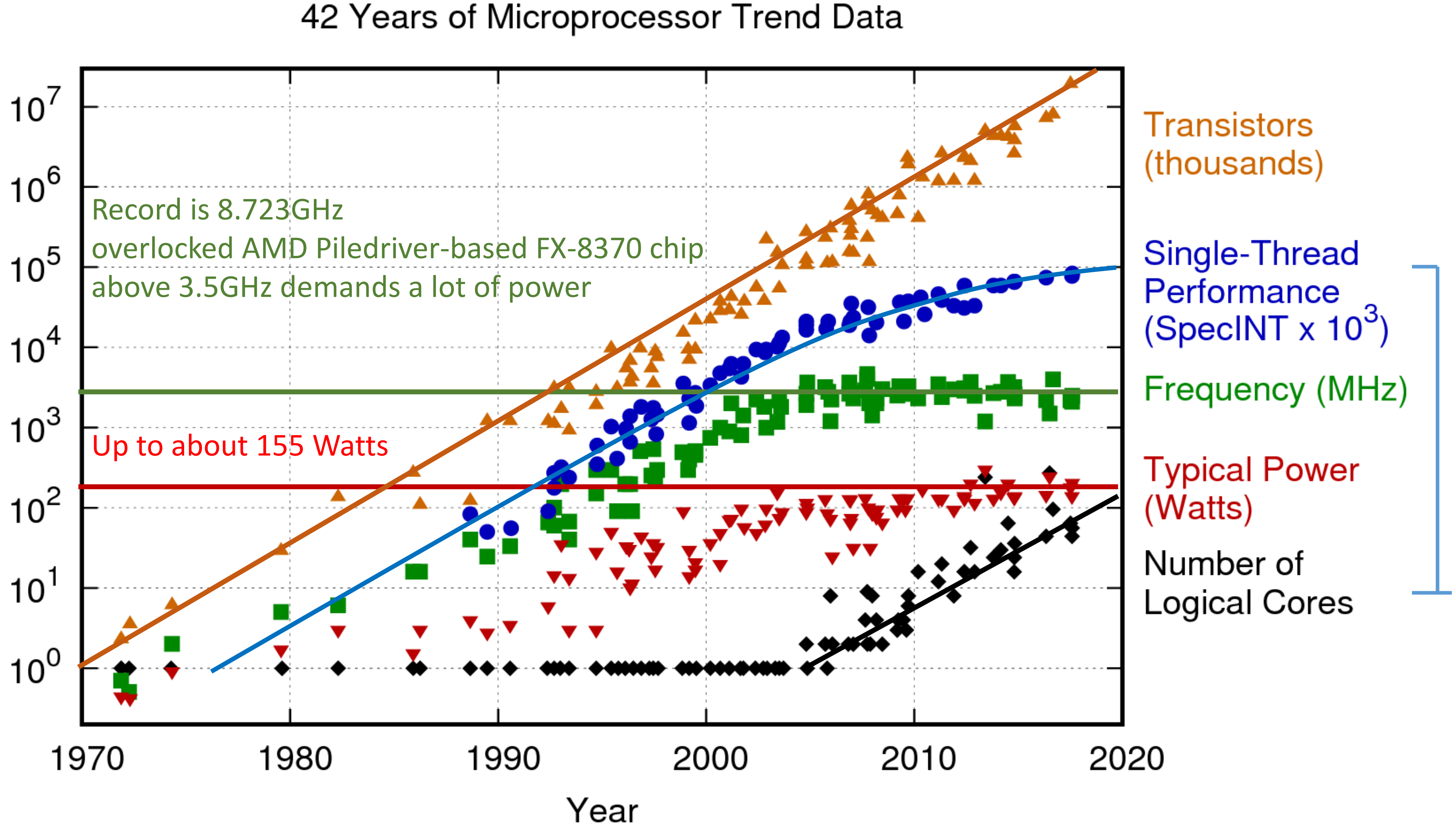
QED Moore's law has worked well so far…

Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Not everything is advancing as quickly as the CPU….



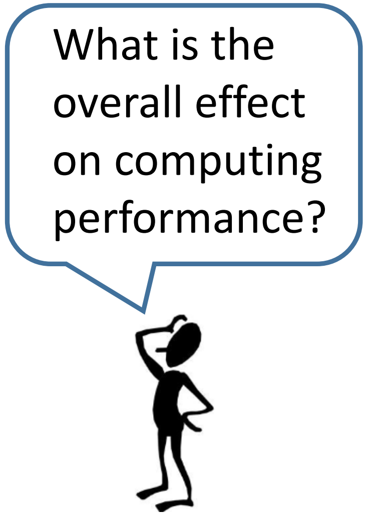
How do we maintain the overall growth in speed when some technologies become a constraint on the system?



Record is 8.723GHz
overlocked AMD Piledriver-based FX-8370 chip
above 3.5GHz demands a lot of power

Up to about 155 Watts

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp



Frequency limits

Power limits

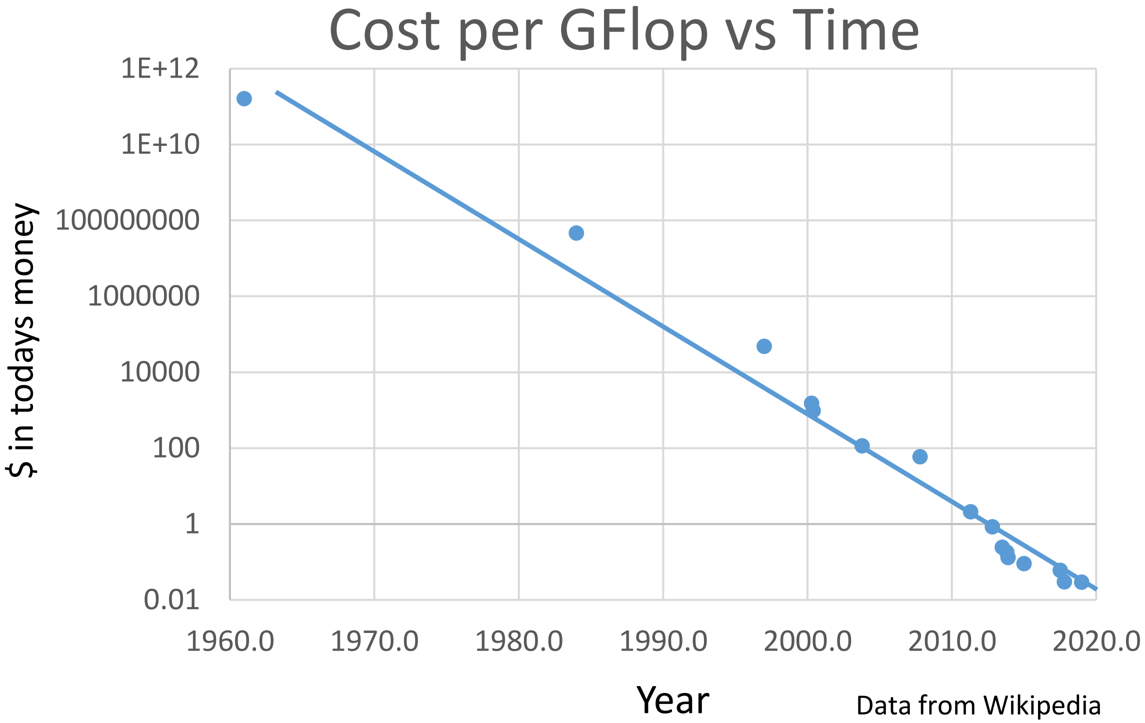
Going parallel is keeping us on the curve.

https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

More on this later in the module…

FLOPS

are units of measure for the numerical computing performance



Data from Wikipedia

How do you go fast?

1: Up the clock rate
2: Improve the cooling
3: Reduce the dissipation – lower voltages
4: Do things in parallel

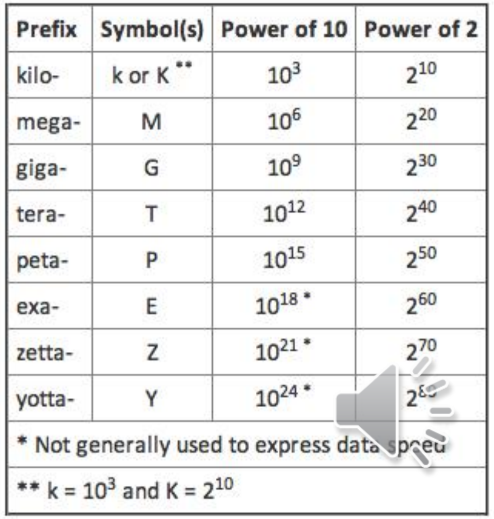
5: Algorithmic changes such as quick search or FFT

1 bricklayer can build a wall in 3 days, but 3 three bricklayers can't build a wall in 1 day (because the cement needs time to set between courses).
Data dependences can affect parallel operation.

FLOP is defined as either an addition or multiplication of single (32 bit) or double (64 bit) precision numbers in conformance with the IEEE 754 standard.

GPU GTX980 4.6 TFlop/s,
Processor i7-5930 336 GFlop/s,

4.6x10^{12}/336x10^{9}=13.7 (GPU is faster, note in both cases speeds in practice are lower)



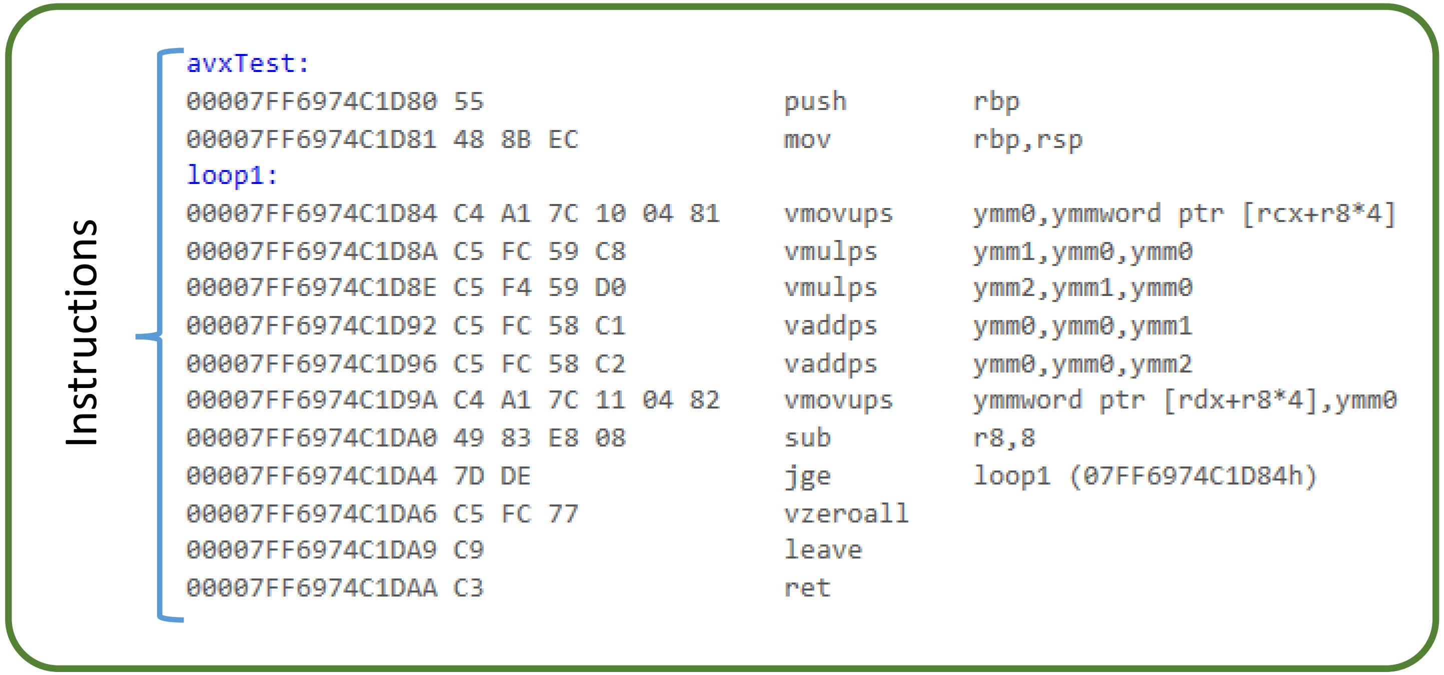
Prefix	Symbol(s)	Power of 10	Power of 2
kilo-	k or K **	10^3	2^{10}
mega-	M	10^6	2^{20}
giga-	G	10^9	2^{30}
tera-	T	10^{12}	2^{40}
peta-	P	10^{15}	2^{50}
exa-	E	10^{18} *	2^{60}
zetta-	Z	10^{21} *	2^{70}
yotta-	Y	10^{24} *	2^{80}

* Not generally used to express data speed

** k = 10^3 and K = 2^{10}

MIPS

Instructions per second (IPS) is a measure of a computer's processor speed. For CISC computers different instructions take different amounts of time, so the value measured depends on the instruction mix; even for comparing processors in the same family the IPS measurement can be problematic. Many reported IPS values have represented "peak" execution rates on artificial instruction sequences with few branches and no cache contention, whereas realistic workloads typically lead to significantly lower IPS values.



```
avxTest:
00007FF6974C1D80 55                     push        rbp
00007FF6974C1D81 48 8B EC               mov         rbp,rsp
loop1:
00007FF6974C1D84 C4 A1 7C 10 04 81      vmovups     ymm0,ymmword ptr [rcx+r8*4]
00007FF6974C1D8A C5 FC 59 C8            vmulps      ymm1,ymm0,ymm0
00007FF6974C1D8E C5 F4 59 D0            vmulps      ymm2,ymm1,ymm0
00007FF6974C1D92 C5 FC 58 C1            vaddps      ymm0,ymm0,ymm1
00007FF6974C1D96 C5 FC 58 C2            vaddps      ymm0,ymm0,ymm2
00007FF6974C1D9A C4 A1 7C 11 04 82      vmovups     ymmword ptr [rdx+r8*4],ymm0
00007FF6974C1DA0 49 83 E8 08            sub         r8,8
00007FF6974C1DA4 7D DE                  jge         loop1 (07FF6974C1D84h)
00007FF6974C1DA6 C5 FC 77               vzeroall
00007FF6974C1DA9 C9                     leave
00007FF6974C1DAA C3                     ret
```

i7 – 3770K – 106,924 MIPS

This processor can do 32x 32 bit (1024 bits) floating point calculations per instruction cycle.

4 cores (8 threads) each core can manage a 256 bit register (e.g. ymm0) 4x256=1024.

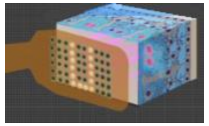
So 32 floating numbers per instruction cycle across 4 cores, 32(float calculations)x 1.06924x10^{11} (IPS) = 3,420 G-flops over estimate

(153.5 Gflops reported => 1.43 flops / ips)

Topic 1.2: Types of computer

There are many ways that we could choose to compute

IBM TrueNorth

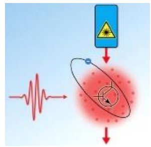


Retinal Prosthesis

Neurones

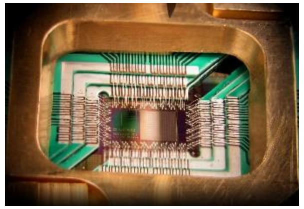
Spatial filtering

Holography



Light

Optical transistor



D-Wave 128 Qubits

n bits can be in only one of 2^n states

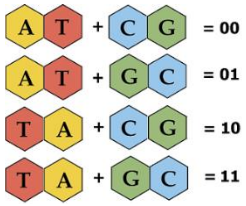
n Qubits contains superposition of 2^n states

Quantum

Difference Engine

Pin Wheel

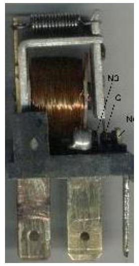
Mechanical



Coding of data

DNA

Structure



Transistor

Relay

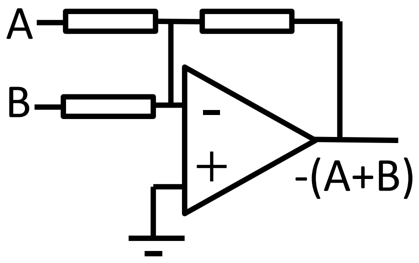
Valve

Electrical

IC

The DEC (Digital Electronic Computer) is by far the most popular

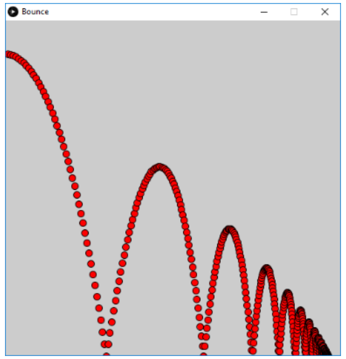
Analogue



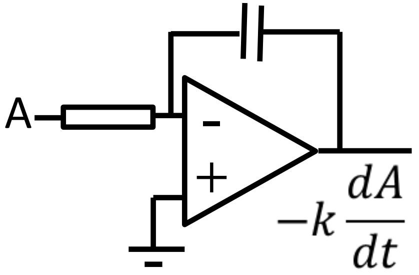
Analog adder circuit

$$-mg + kv + m\frac{d^2y}{dt^2} = 0$$

Differential Equation



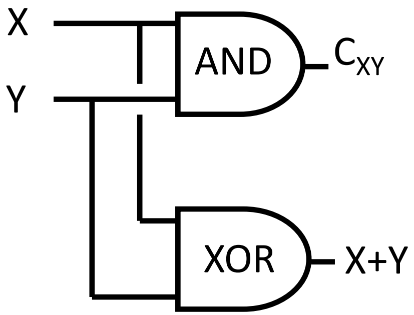
Response



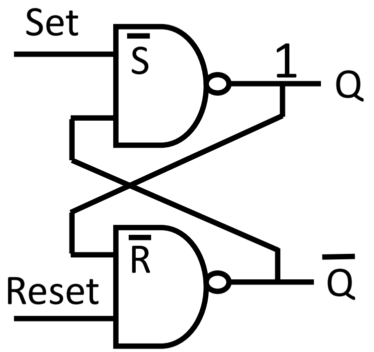
Differentiator

An analogue signal is continuous (can have all values in a range). The signal is processed by electronics that can amplify, multiply, differentiate and integrate input signals to produce an output. Projectile motion can be simulated using a circuit that responds in the same way as the differential equation describing the motion .

Digital

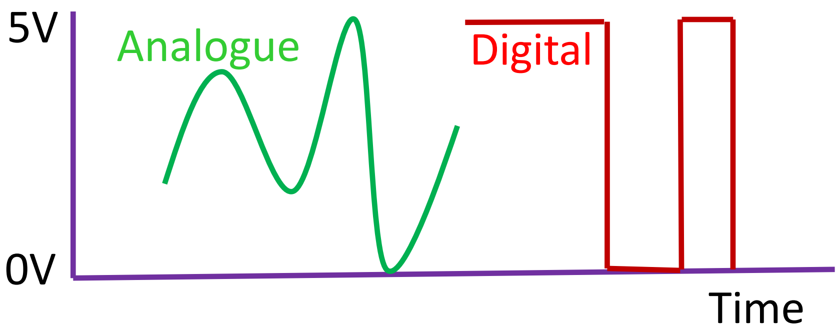


Adder



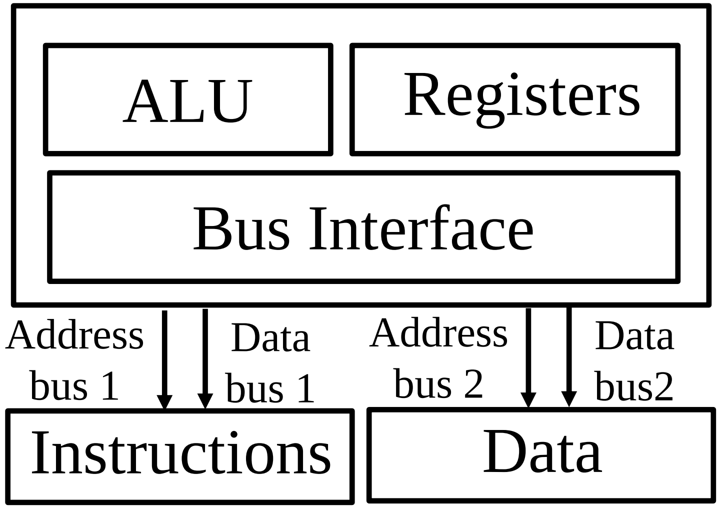
RS Flip flop (1 bit memory)

A digital signal is discrete (can have one of two values in a range). Digital logic can use 0 volts to represent 0 in binary and say 5 volts to represent 1 in binary. Digital circuits to process the 1 or 0 information are much easier to build as they only need to switch between two values rather have an output that is continuous and accurate over a range.



Types of computer

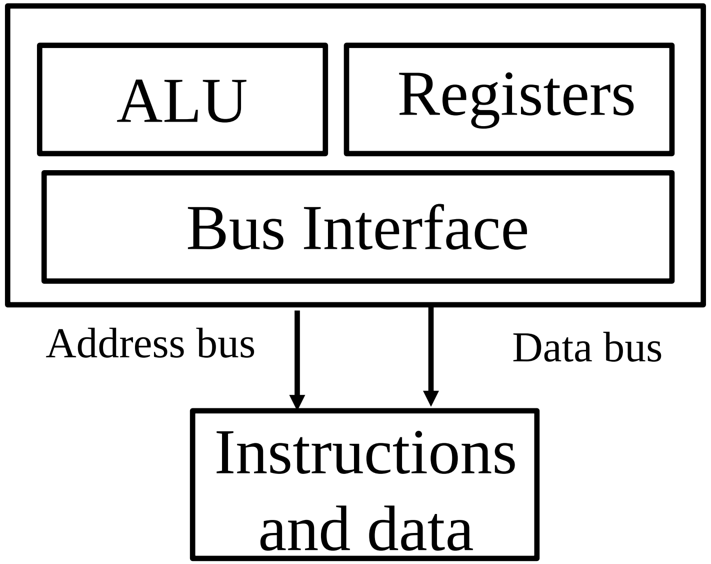
Harvard



```
Arizona PIC Microcontroller
Instruction   Data   Assembly language
5A            01    movlw B'00000001' ; w=1
4F            03    movwf PORTB   ; Port B=w
```

Memory and data buses to store instructions (operators) is separate to the memory used for data (operands).

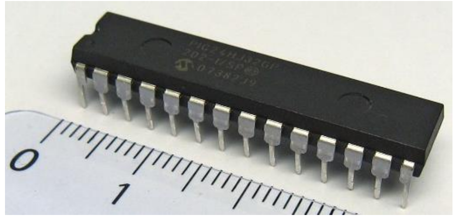
Von Neumann



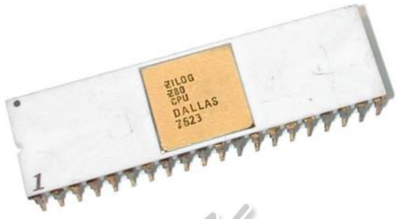
```
x86 Assembly language
Instruction and Data            Assembly language
B0 00                              mov al,0
B8 02 38                           mov ax,568
```

There is only one memory used for both data and instructions. Looking at bytes in memory it would difficult to tell with certainty which stores code and which stores data.

John von Neumann
1903 –1957



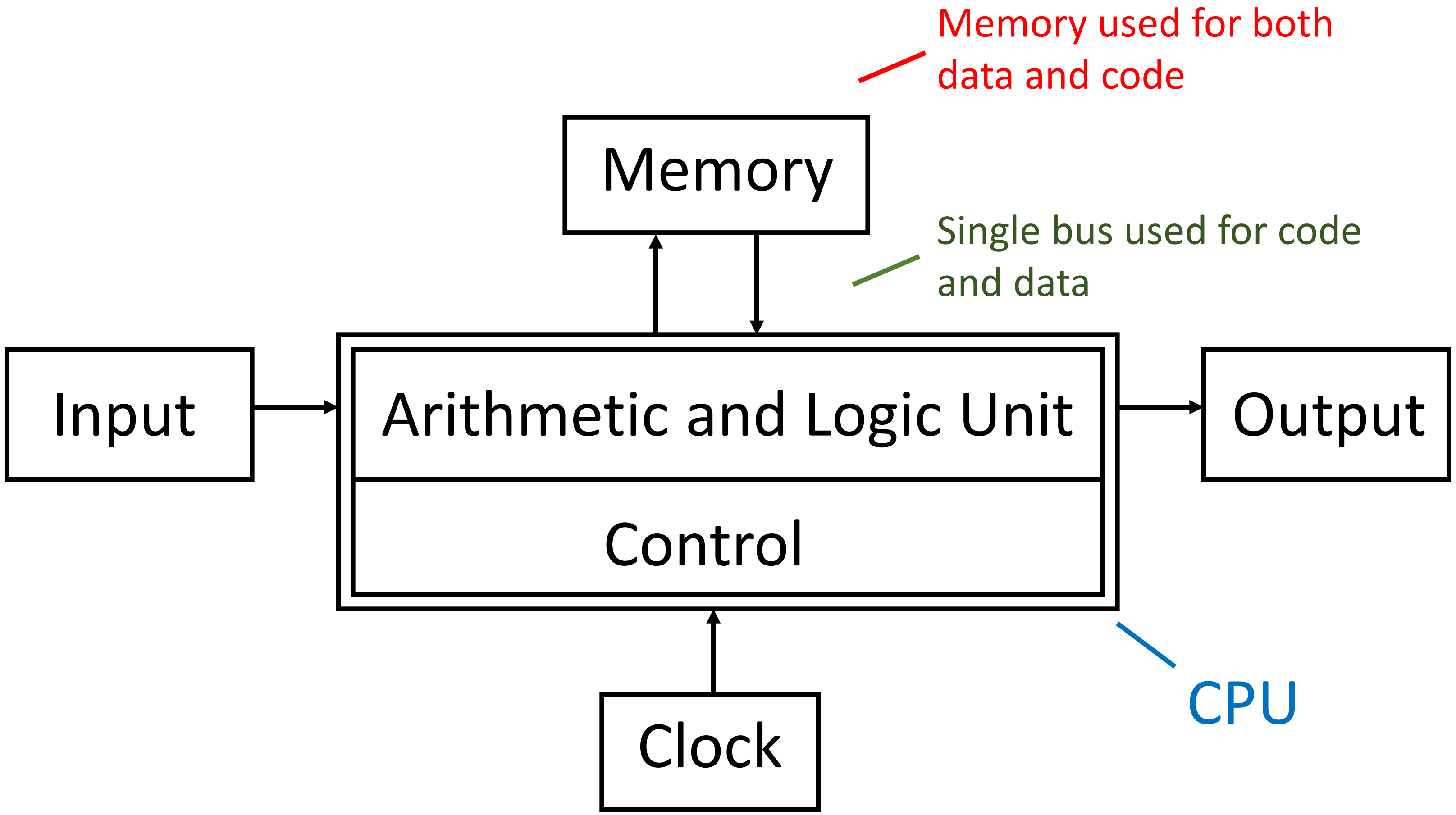
Arizona PIC - Harvard



Zilog Z80A - Von Neumann

Even though the x86 processor is von Neumann, internally it can split data and instruction pipelines to speed things up (essentially Harvard in nature)

Organisation of a von Neumann digital electronic computer



Memory used for both data and code

Memory

Single bus used for code and data

Input

Arithmetic and Logic Unit

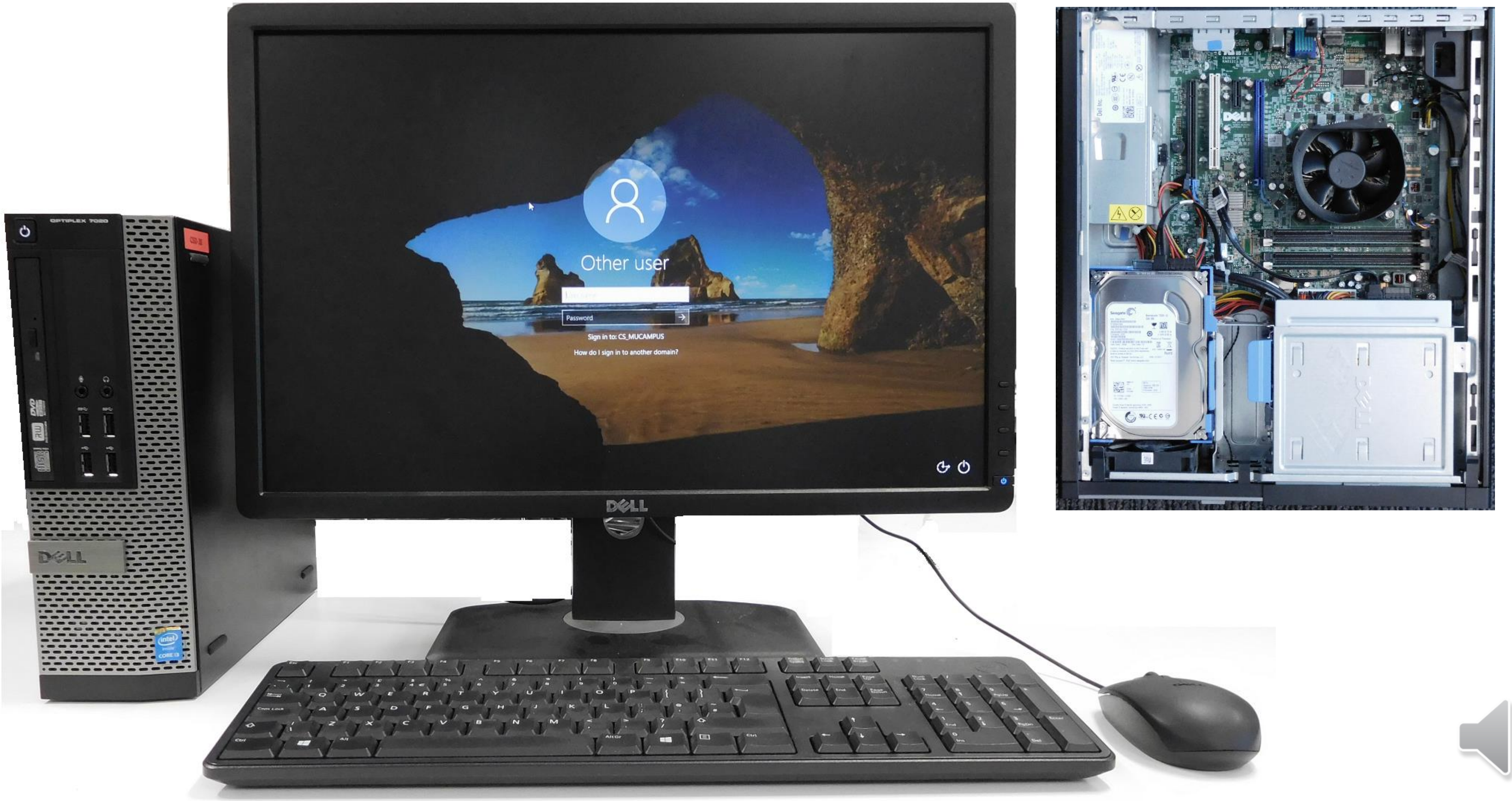
Control

Output

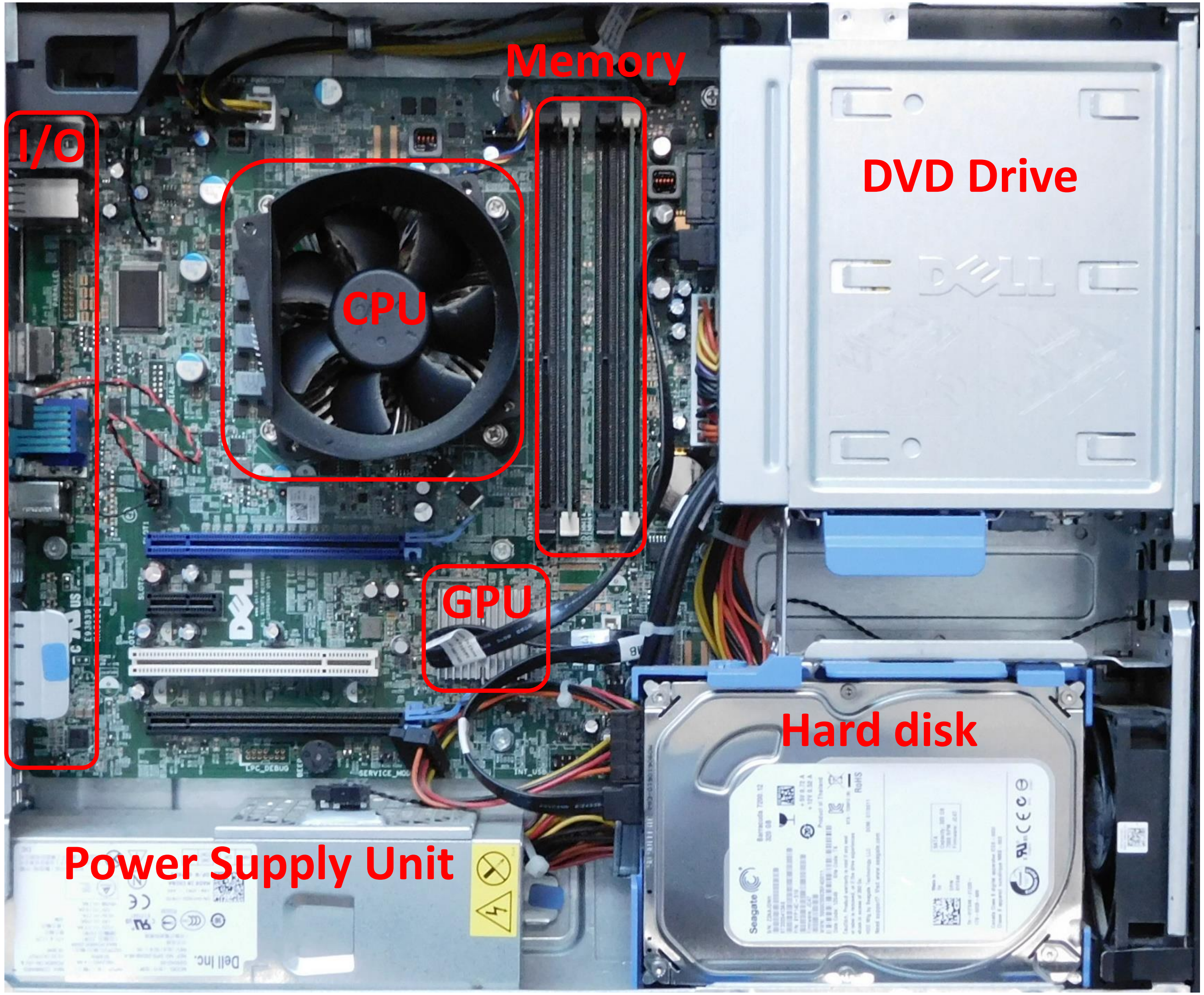
Clock

CPU

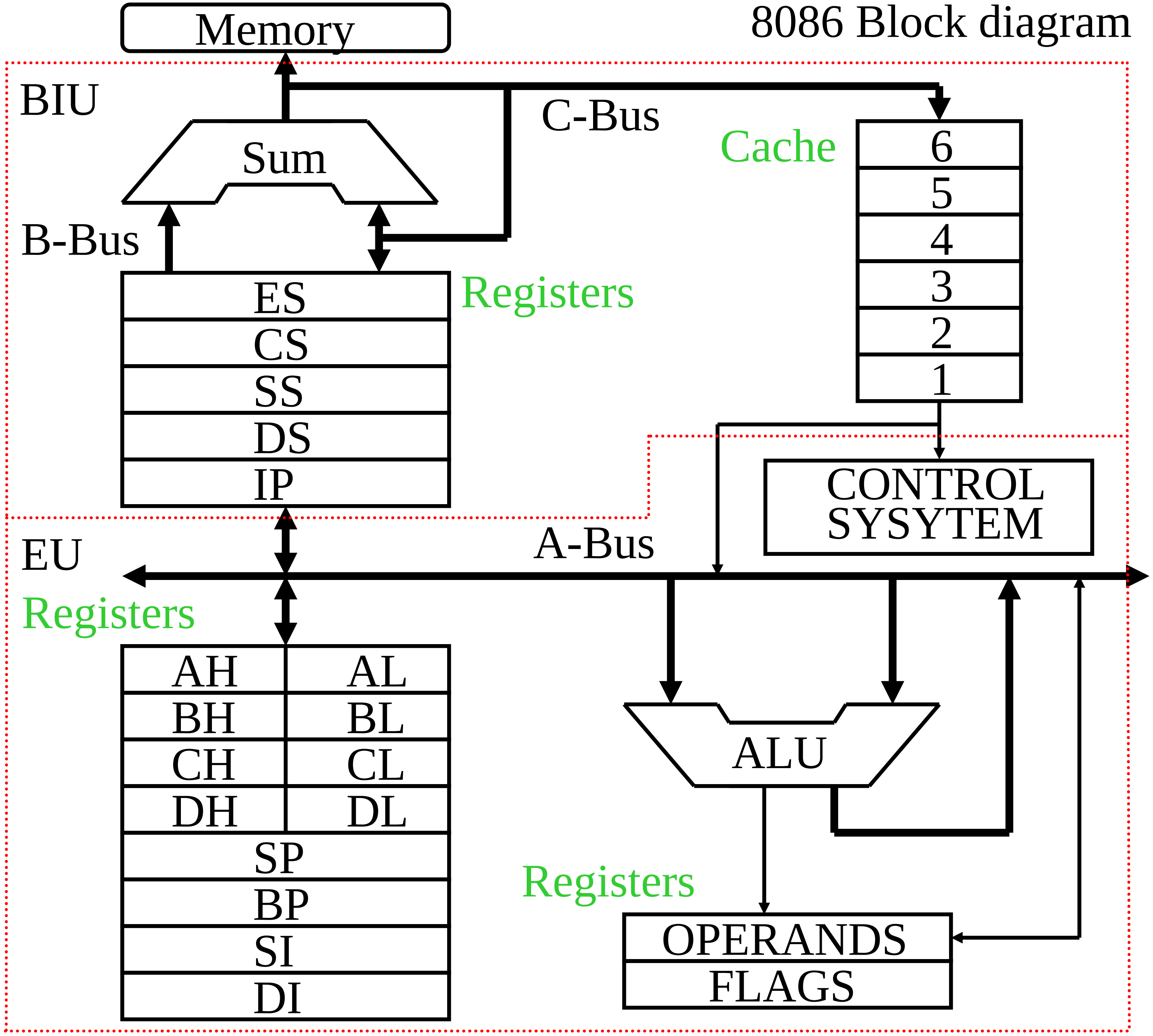
Topic 1.3: Lift the lid on a computer



PC Mother Board



8086 Block diagram



BIU: Bus Interface Unit
The BIU sends outs addresses fetches instructions from memory and reads and writes to ports and to memory.

EU: Execution Unit
The EU instructs the BIU where to fetch instructions, it decodes instructions and executes instructions.

The EU contains both the ALU and Control Circuitry.

The A-Bus, B-Bus and C-Bus are high speed data paths contained within the Microprocessor itself.

The control unit fetches instructions from the queue.

The queue is a first in first out store of 6 bytes.

The store is kept full by the BIU. This means that main memory is not accessed for each byte of each instruction. The technique is known as pipleling.

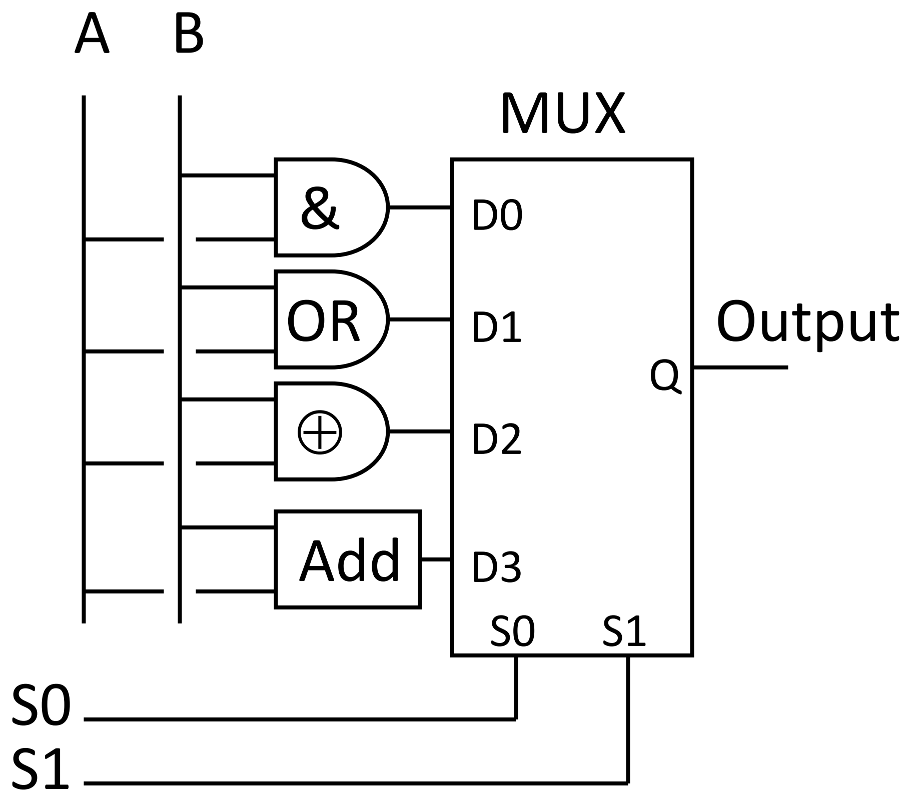
Some instructions such as conditional jumps and call to subroutines can not be pipelined (more later).

Topic 1.4: The ALU (EU)

MUX: Multiplexer

The ALU (Arithmetic Logic Unit) in the 8086 can ADD, Subtract, AND, OR, XOR, increment, decrement, complement and shift 16-bit binary numbers.

Add:	Ouptut=A+B
Subtract:	Output=A-B
Exor:	Output=A⊕B
AND:	Output=A.B
OR:	Output=A+B
Pass:	Output=A
Complement:	Output=$\overline{A}$
Set:	Output=1
Shift Left	Output=A*2
Shift Right	Output=A/2

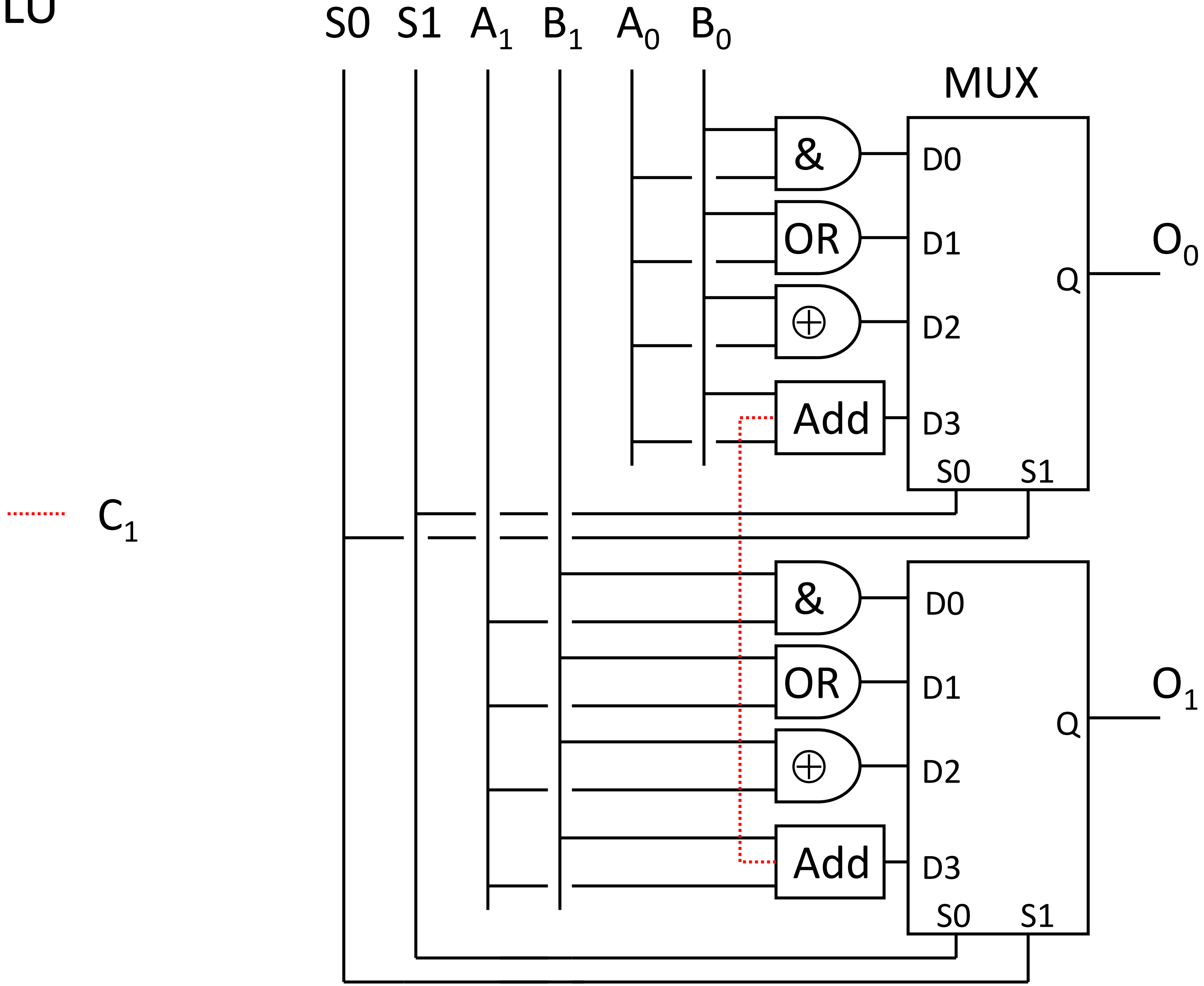


Inputs A and B are operated on by all the functions available in parallel.

The multiplexer connects the ALU output to the desired function.

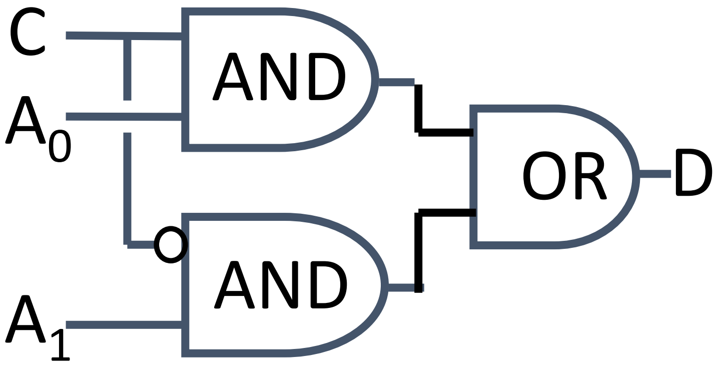
S0	S1	Output
0	0	A&B
0	1	AORB
1	0	A⊕B
1	1	A+B

Two bit ALU


S0
S1
A_1
B_1
A_0
B_0
MUX
&
OR
⊕
Add
D0
D1
D2
D3
Q
O_0
S0
S1
C_1
&
OR
⊕
Add
D0
D1
D2
D3
Q
O_1
S0
S1

Multiplexers

The Control line C selects which input is routed to the output.



C	D
0	A_1
1	A_0

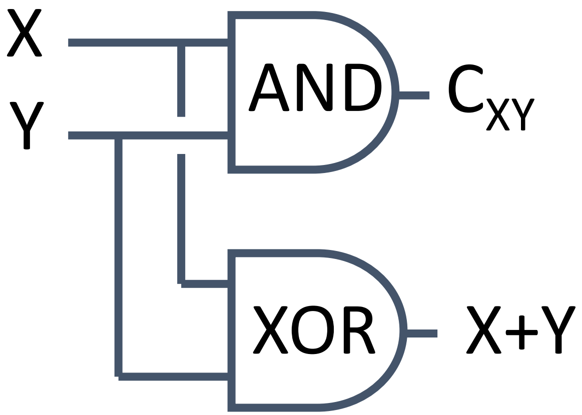
AND

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

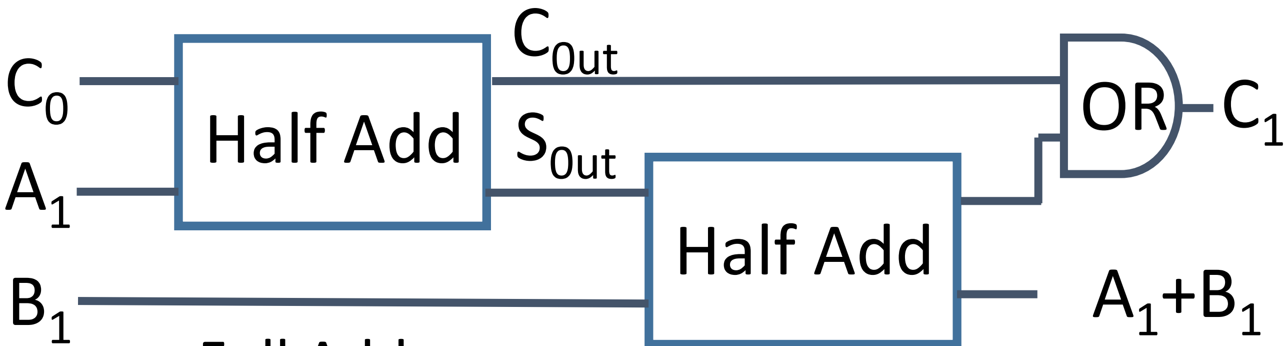
OR

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Adders



Half Adder



Full Adder

Note: The half adder can not take carry in bits.

Combining two half adders creates a full adder capable of accepting carry in.

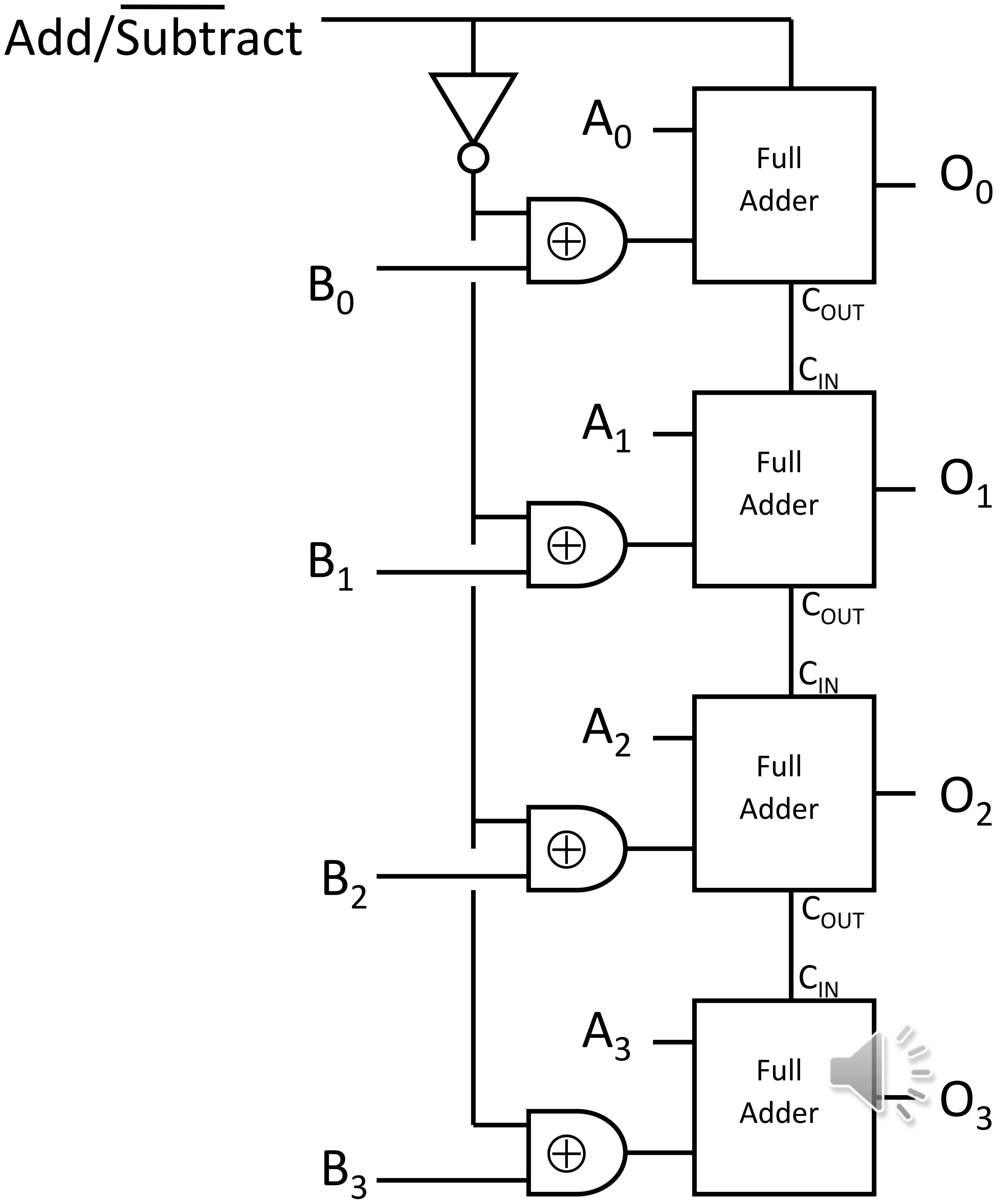
4 Bit Adder/Subtract

Twos Complement of B is generated when AS line is low.

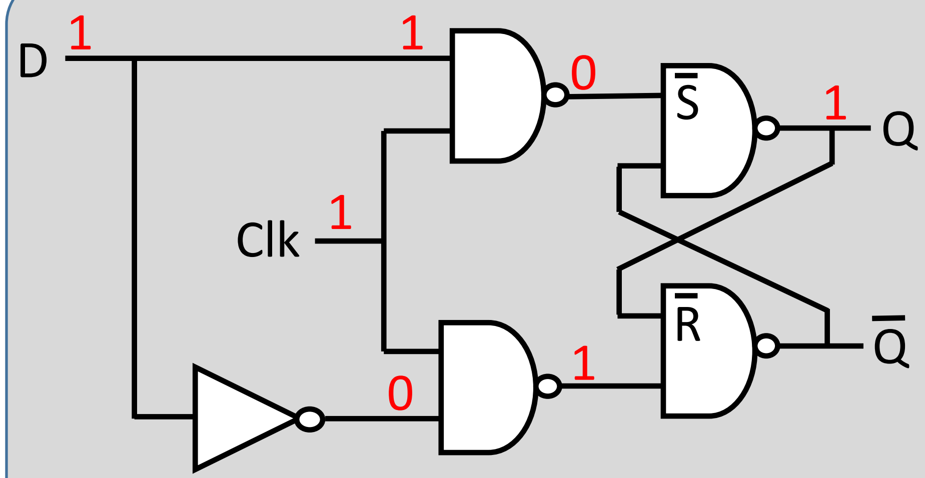
Adding twos complement of a number gives the same result as subtracting the number.

When Add is high B is unchanged and the carry in bit to the first full adder is zero. The result is normal addition.

A B	$A \oplus B$
0 0	0
0 1	1
1 0	1
1 1	0



Topic 1.5: Registers = latches = D Type Flip Flop

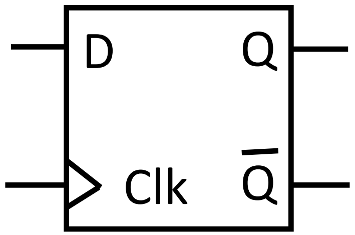


Q=D when Clk=1

Q=last D when Clk=0

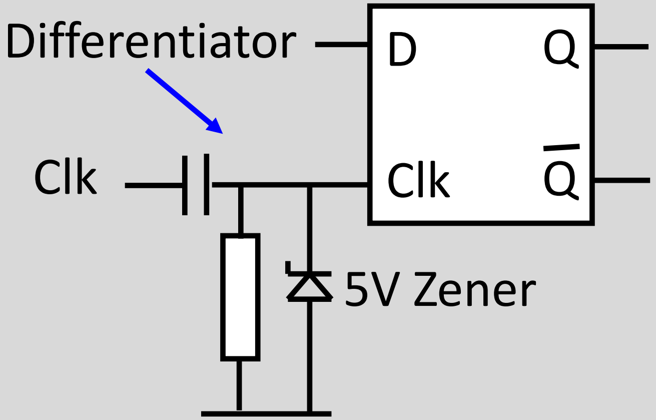
Data (D)=1, Clock=1, Output (Q)=1

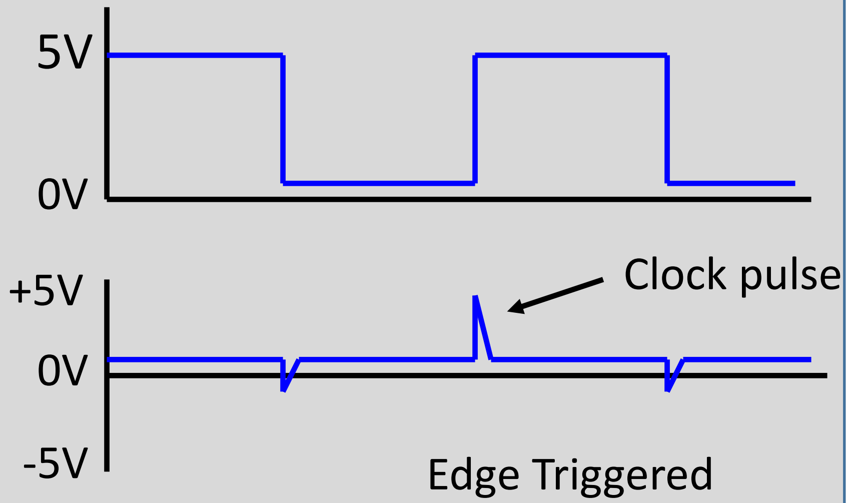
Level Triggered



Clk	D	Q
0	x	No change
1	x	No change
↑	0	0
↑	1	1
↓	x	No change

Input is transferred (stored) to the output on next rising edge of the clock.





Edge Triggered

Latch becomes a Register

The CPU contains a number of internal memory locations made of D type latches.

Each memory location contains a binary number.

The locations have a name and have a specific purpose.

They affect the operation of the hardware.

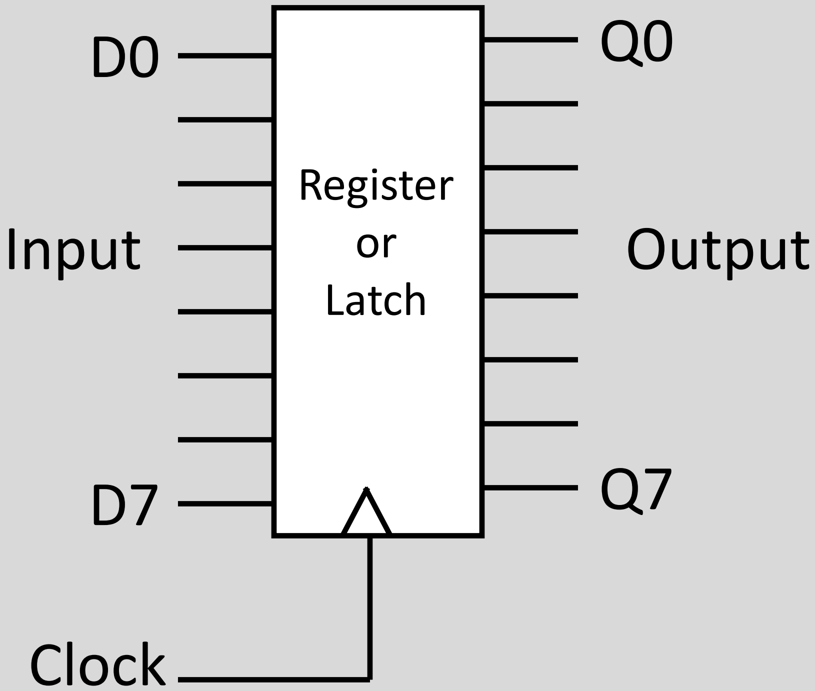
They can be 8, 16, 32, 64, 128 or more bits.

Each register is designed for a specific purpose.

Examples include

Accumulator: Stores running total of a calculation

Instruction pointer: Address of the line of code being executed

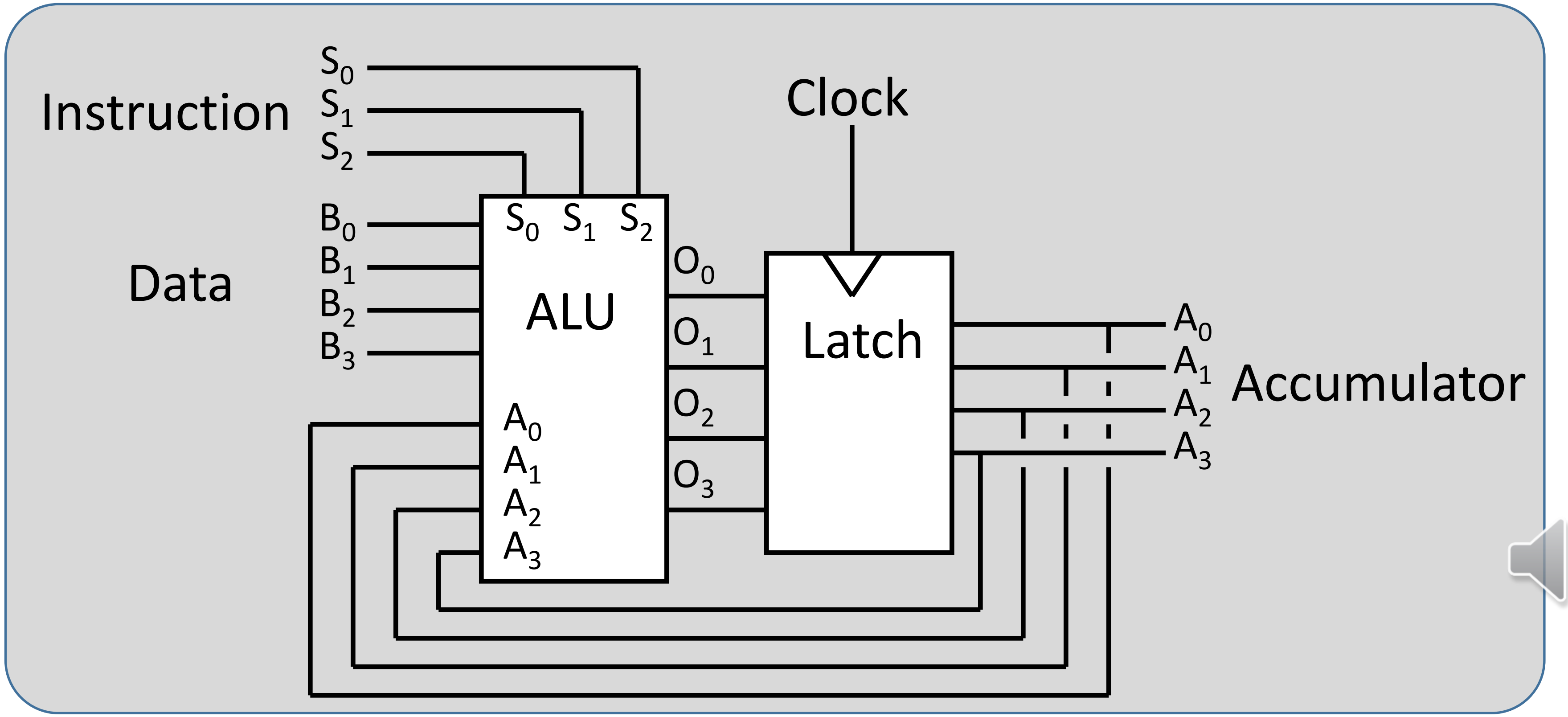


Information (a binary number) is transferred (stored) to the output on next rising edge of the clock.

Topic 1.6: A Simple Calculator

Most Microprocessors have a special register that stores the result of the calculations executed by the ALU, this register is known as the accumulator. The result of a calculation can also be sent to the stack, other registers or even machine memory.

Using the contents of the accumulator as the input to the ALU creates a device that can do complex calculations.

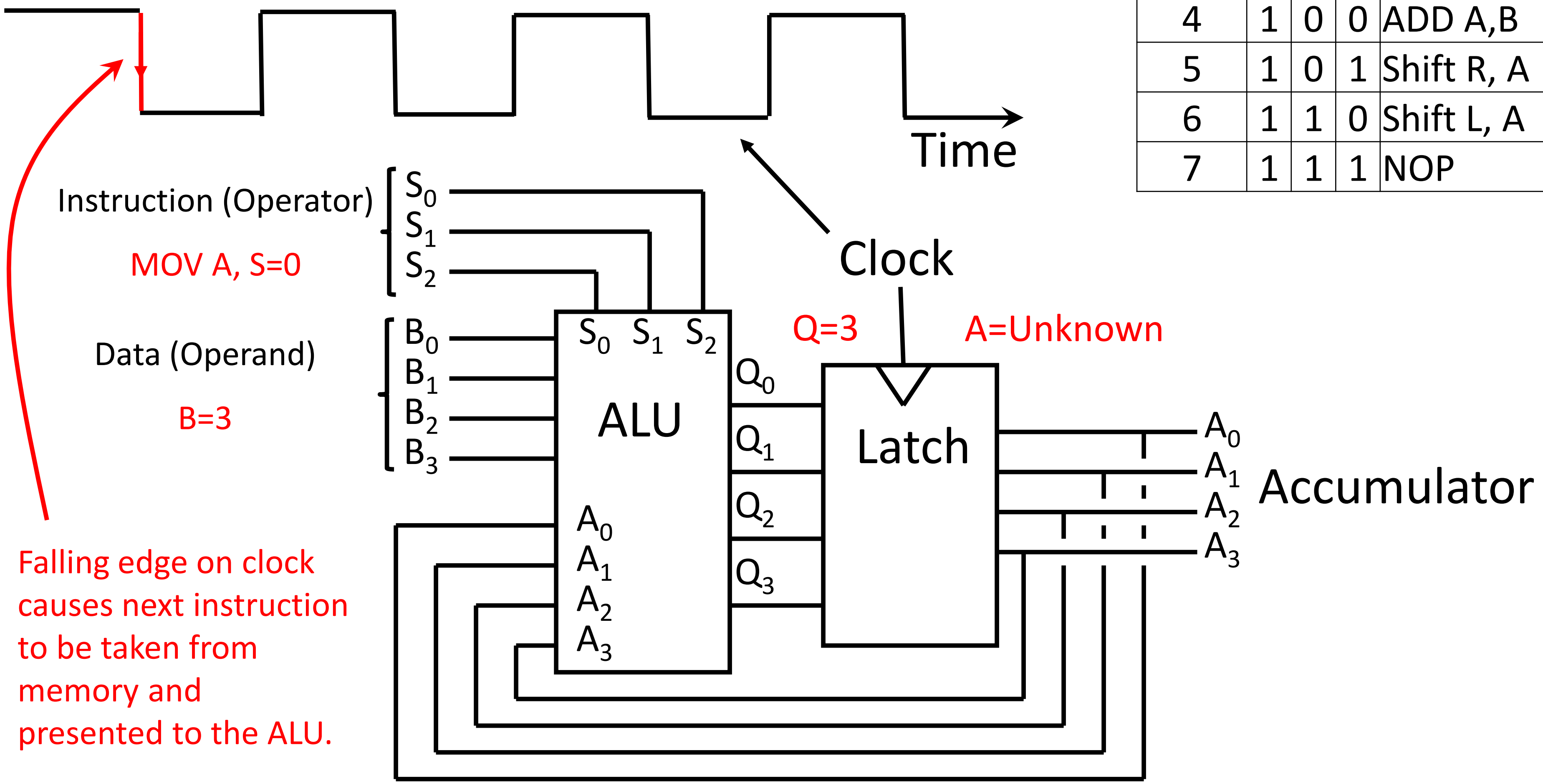


Program to evaluate 5-3

Assembly	Machine code	Accumulator
MOV A, 3	S=0, B=3	A=0011b=3
XOR A,15	S=3, B=15	A=1100b=12
ADD A,1	S=4, B=1	A=1101b=13=-3 TC
ADD A,5	S=4, B=5	A=0010b=2

ALU Instruction set

S(Hex)	S2	S1	S0	Function
0	0	0	0	MOV A,N
1	0	0	1	AND A,B
2	0	1	0	OR A,B
3	0	1	1	XOR A,B
4	1	0	0	ADD A,B
5	1	0	1	Shift R, A
6	1	1	0	Shift L, A
7	1	1	1	NOP



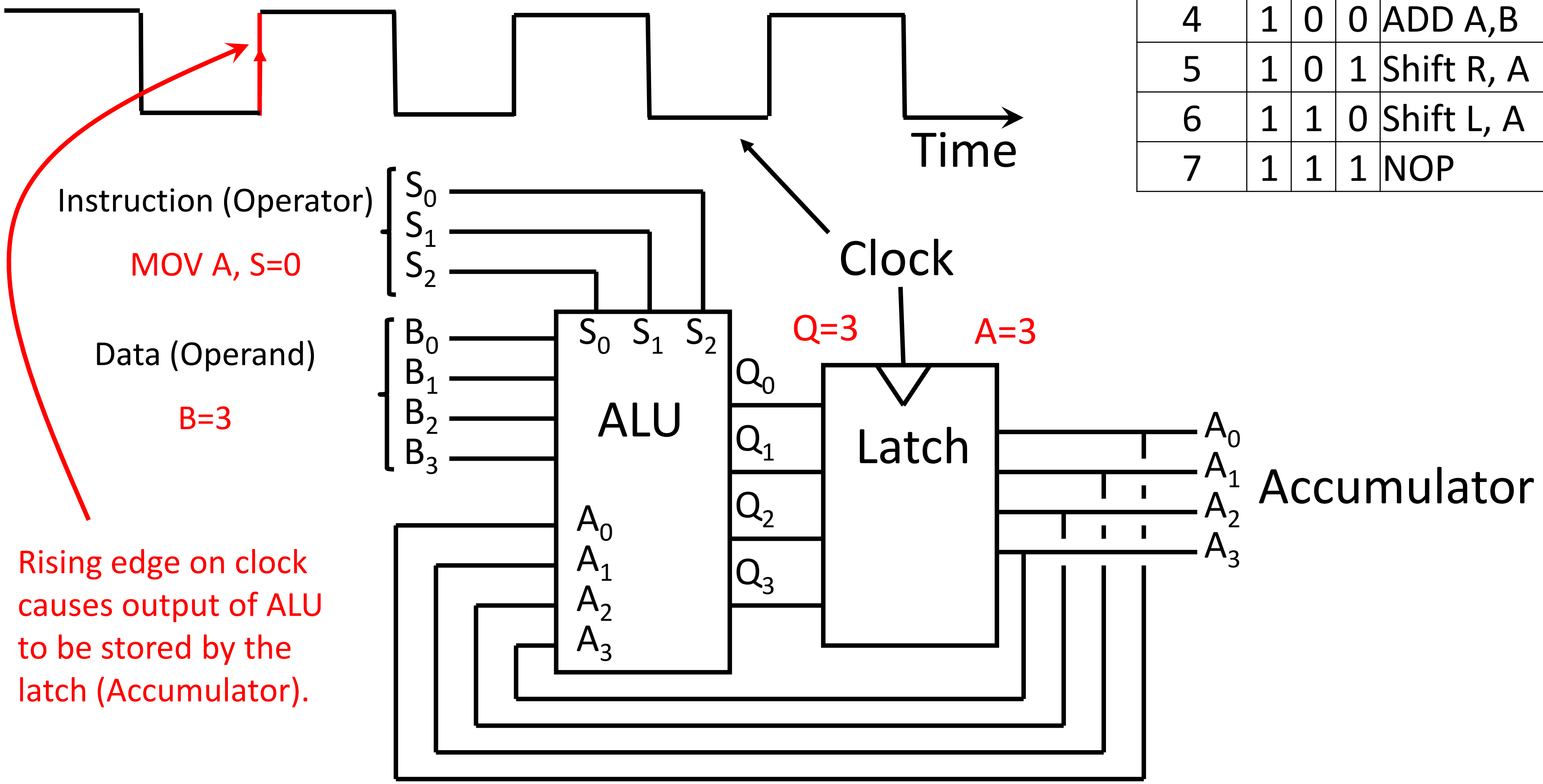
Falling edge on clock causes next instruction to be taken from memory and presented to the ALU.

Program to evaluate 5-3

Assembly	Machine code	Accumulator
MOV A, 3	S=0, B=3	A=0011b=3
XOR A,15	S=3, B=15	A=1100b=12
ADD A,1	S=4, B=1	A=1101b=13=-3 TC
ADD A,5	S=4, B=5	A=0010b=2

ALU Instruction set

S(Hex)	S2	S1	S0	Function
0	0	0	0	MOV A,N
1	0	0	1	AND A,B
2	0	1	0	OR A,B
3	0	1	1	XOR A,B
4	1	0	0	ADD A,B
5	1	0	1	Shift R, A
6	1	1	0	Shift L, A
7	1	1	1	NOP



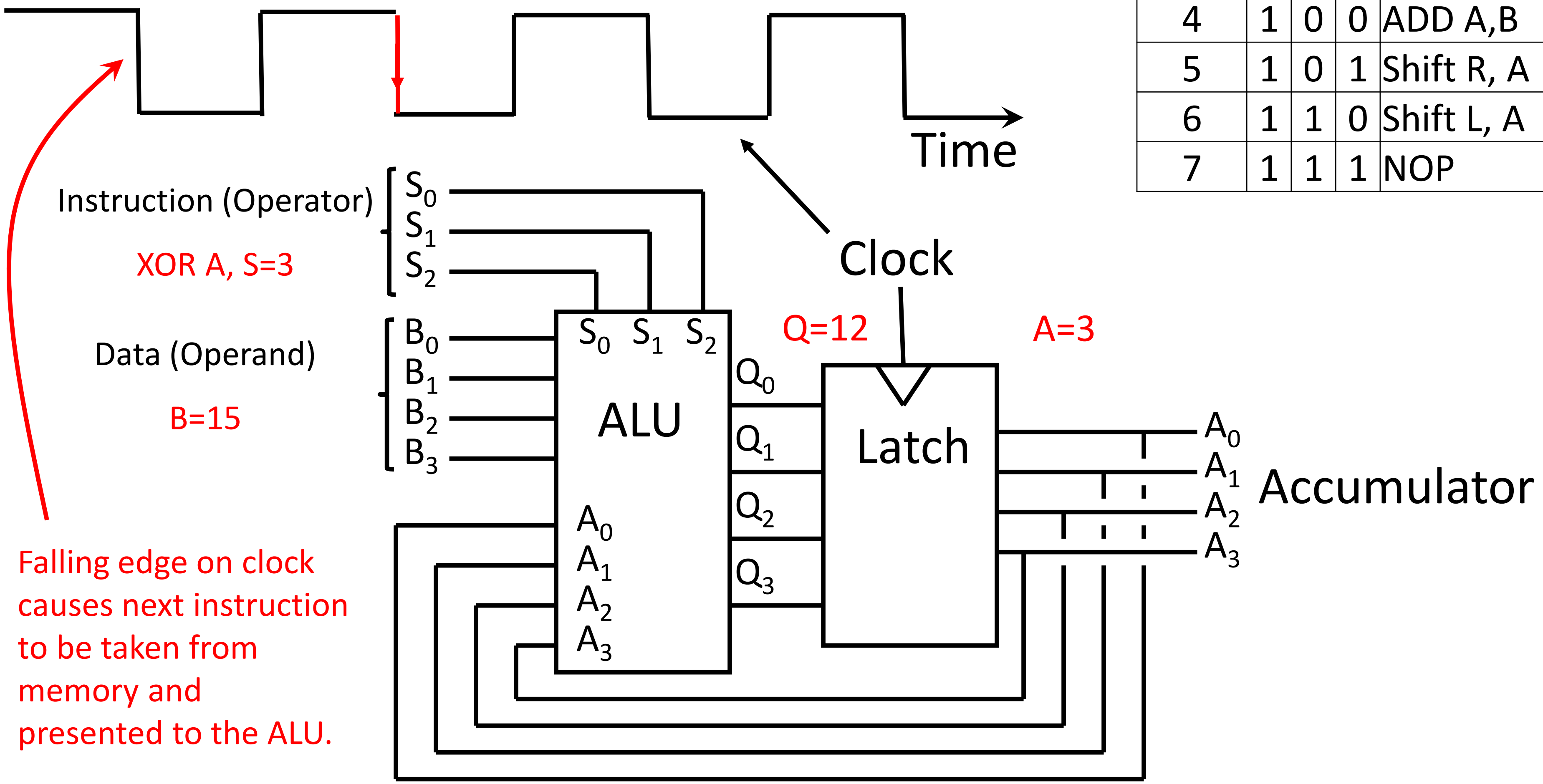
Rising edge on clock causes output of ALU to be stored by the latch (Accumulator).

Program to evaluate 5-3

Assembly	Machine code	Accumulator
MOV A, 3	S=0, B=3	A=0011b=3
XOR A,15	S=3, B=15	A=1100b=12
ADD A,1	S=4, B=1	A=1101b=13=-3 TC
ADD A,5	S=4, B=5	A=0010b=2

ALU Instruction set

S(Hex)	S2	S1	S0	Function
0	0	0	0	MOV A,N
1	0	0	1	AND A,B
2	0	1	0	OR A,B
3	0	1	1	XOR A,B
4	1	0	0	ADD A,B
5	1	0	1	Shift R, A
6	1	1	0	Shift L, A
7	1	1	1	NOP



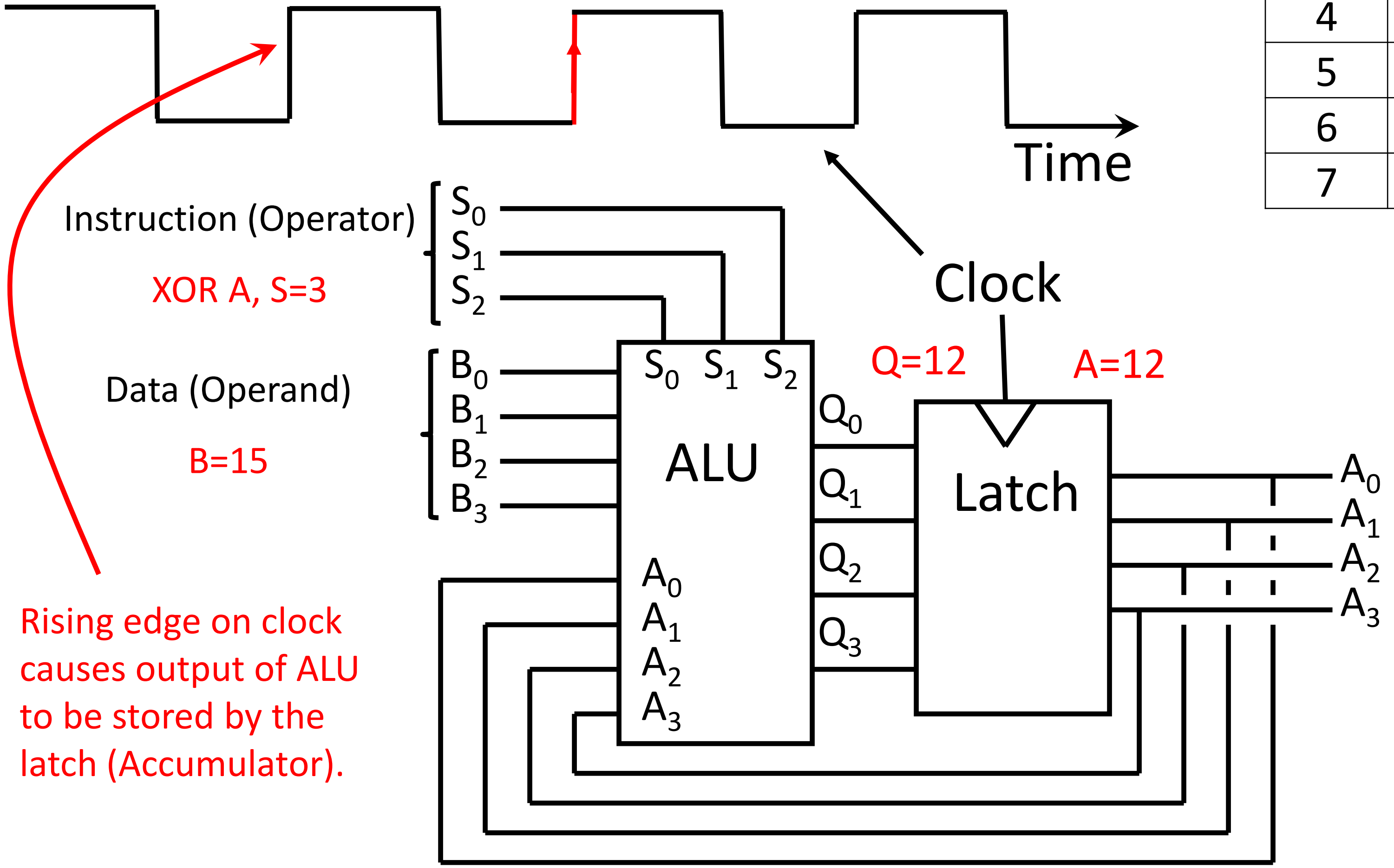
Falling edge on clock causes next instruction to be taken from memory and presented to the ALU.

Program to evaluate 5-3

```
Assembly    Machine code  Accumulator
MOV A, 3    S=0, B=3      A=0011b=3
XOR A,15    S=3, B=15     A=1100b=12
ADD A,1     S=4, B=1      A=1101b=13=-3 TC
ADD A,5     S=4, B=5      A=0010b=2
```

ALU Instruction set

S(Hex)	S2	S1	S0	Function
0	0	0	0	MOV A,N
1	0	0	1	AND A,B
2	0	1	0	OR A,B
3	0	1	1	XOR A,B
4	1	0	0	ADD A,B
5	1	0	1	Shift R, A
6	1	1	0	Shift L, A
7	1	1	1	NOP



XOR A, S=3

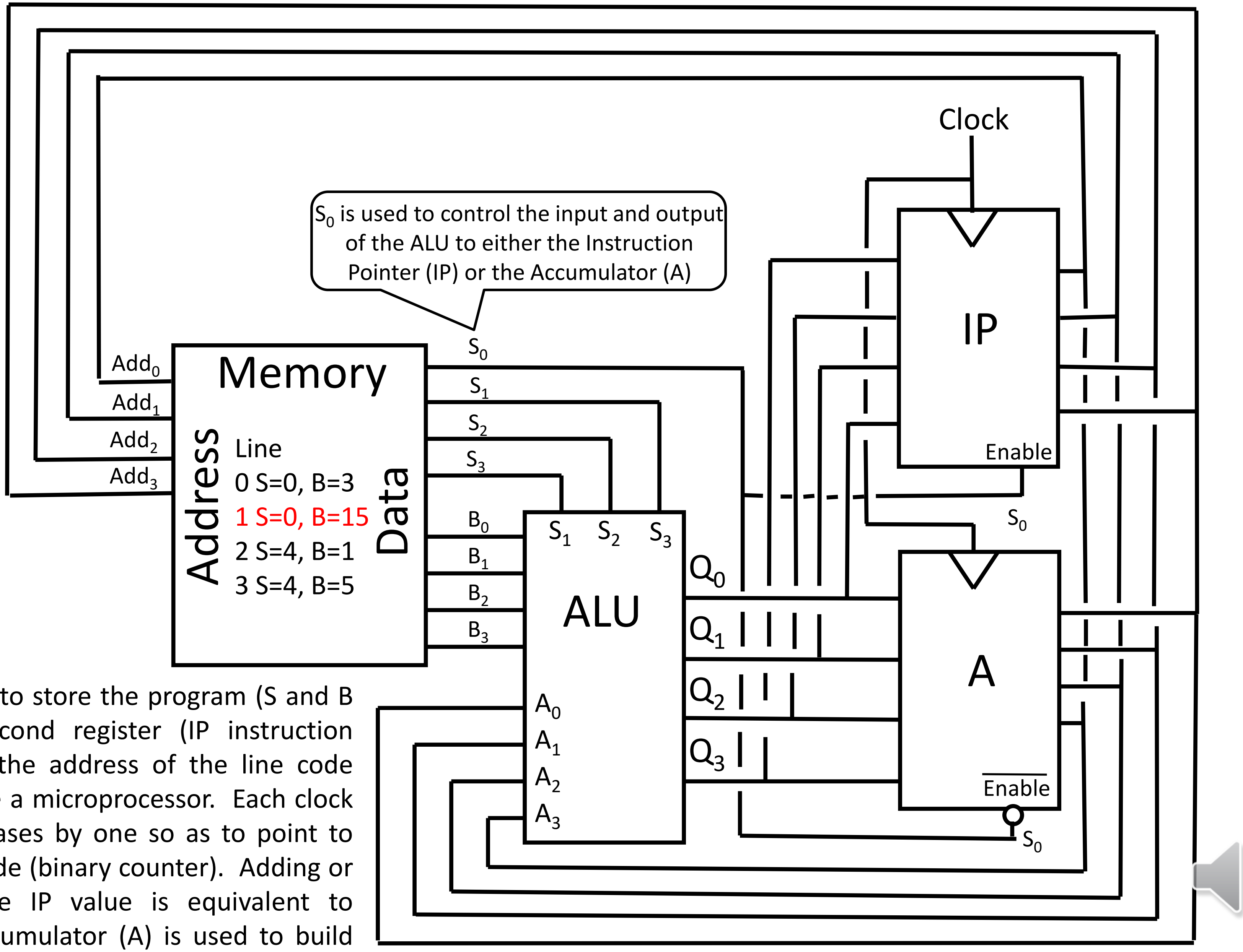
B=15

Accumulator

Rising edge on clock causes output of ALU to be stored by the latch (Accumulator).

And so on…

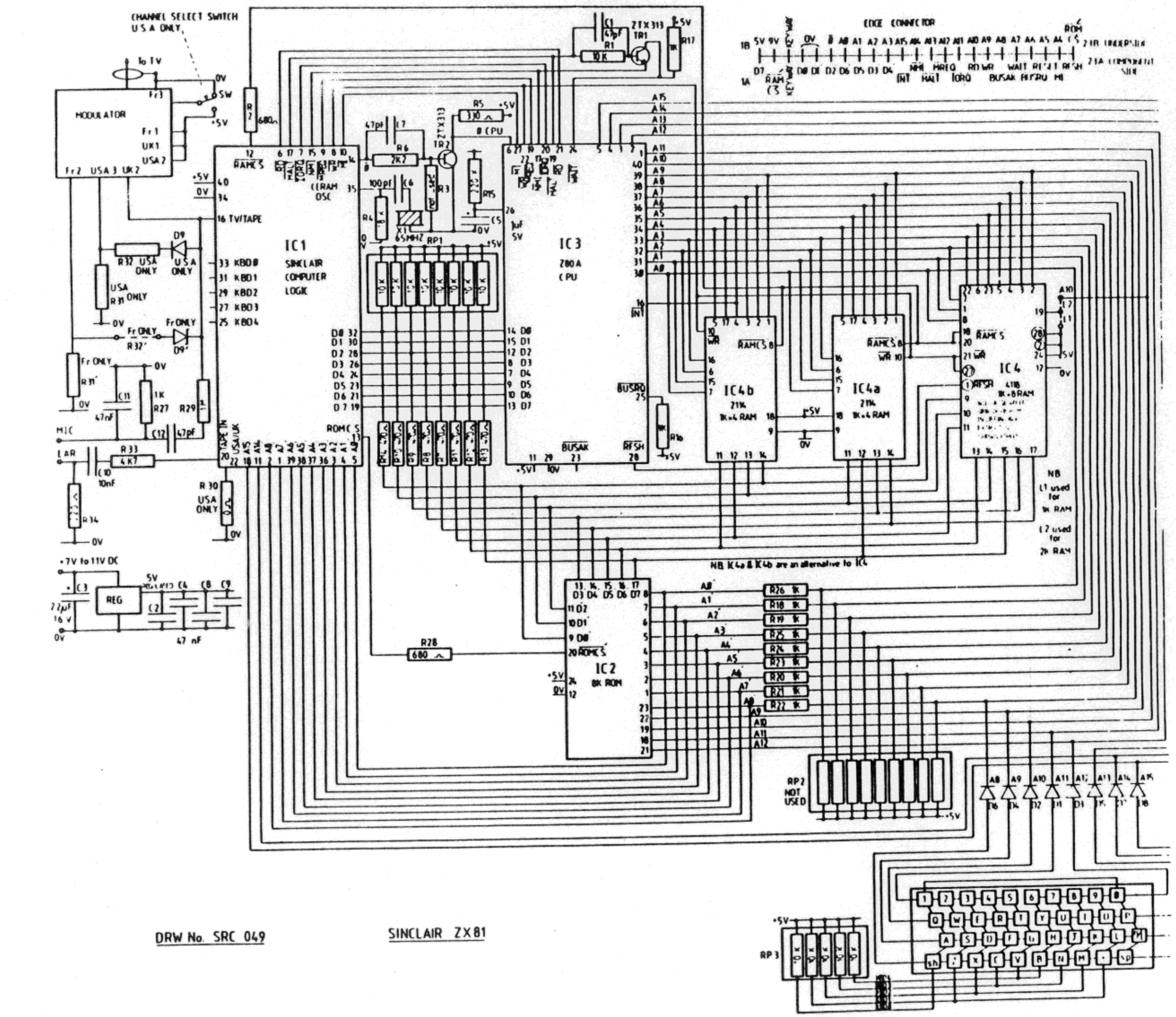
Extending the calculator to become a simple microprocessor

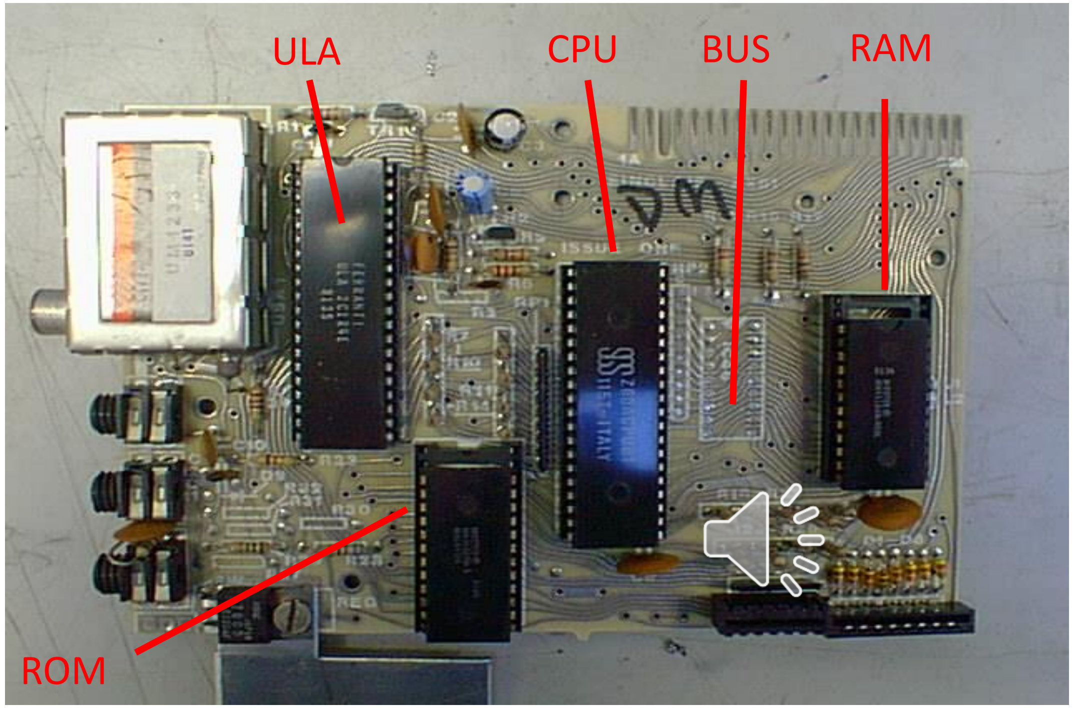


Adding a memory to store the program (S and B values) and a second register (IP instruction pointer) to store the address of the line code allows us to create a microprocessor. Each clock pulse the IP increases by one so as to point to the next line of code (binary counter). Adding or subtracting to the IP value is equivalent to jumping. The Accumulator (A) is used to build the answer. The IP keeps track of the line of code.

Circuit not complete, however it does convey the concept

ZX81 Based Microcomputer



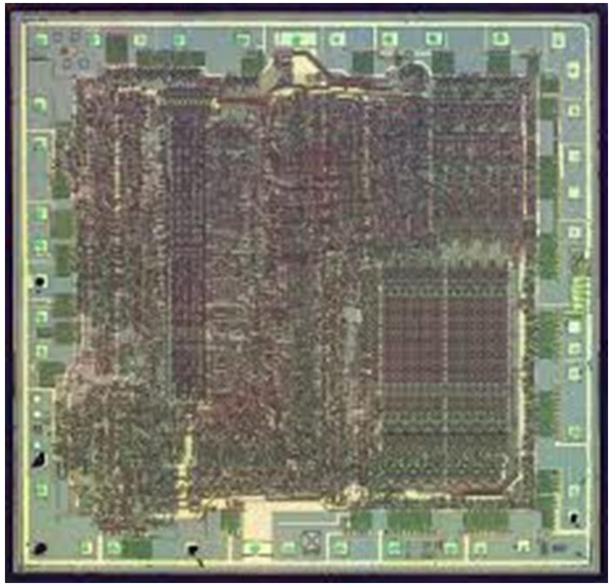


As straightforward as it gets, similar features to a PC.

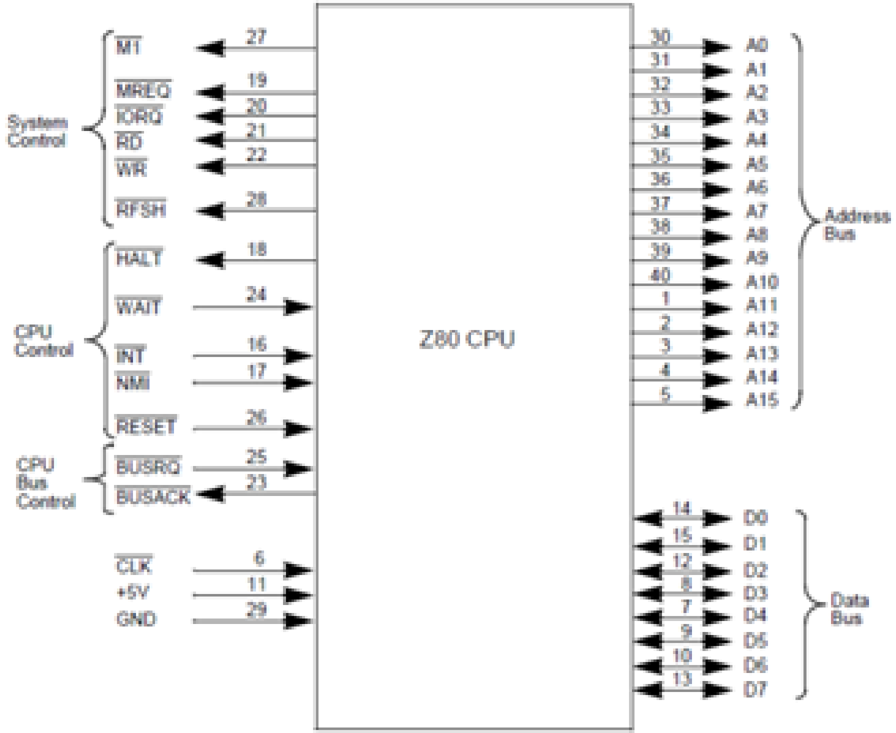
They are all DEC (Digital Electronic Computers)

Topic 1.7: The machine cycle - Z80A Microprocessor example

Package



Die



Circuit diagram symbol

Transistor Count: 8,500
Date: 1976
Manufacturer: Zilog
Feature Size: 4 µm
Area: 18 mm²
Clock Frequency: up to 20 MHz
Registers: 208 bits (6 x 8-bit)

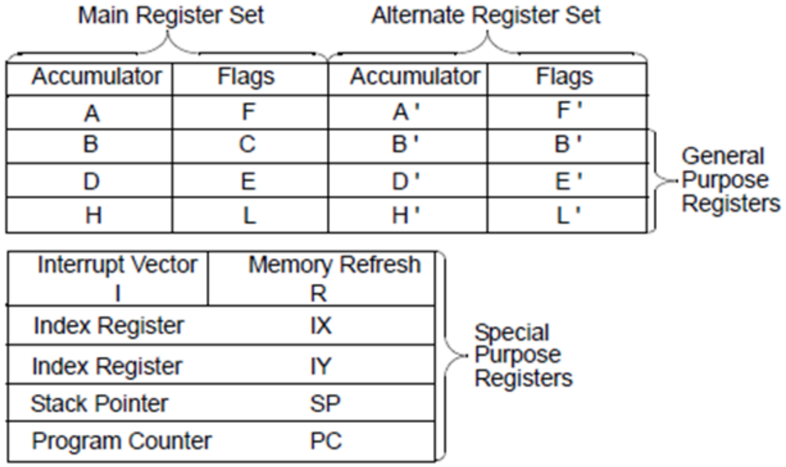
Buses

Address bus: Output from uP, used to address or “wake up” a device.

Data bus: Bidirectional bus, carries data between CPU and Device.

Control bus: Synchronises the data transfer between devices.

Z80 Architecture



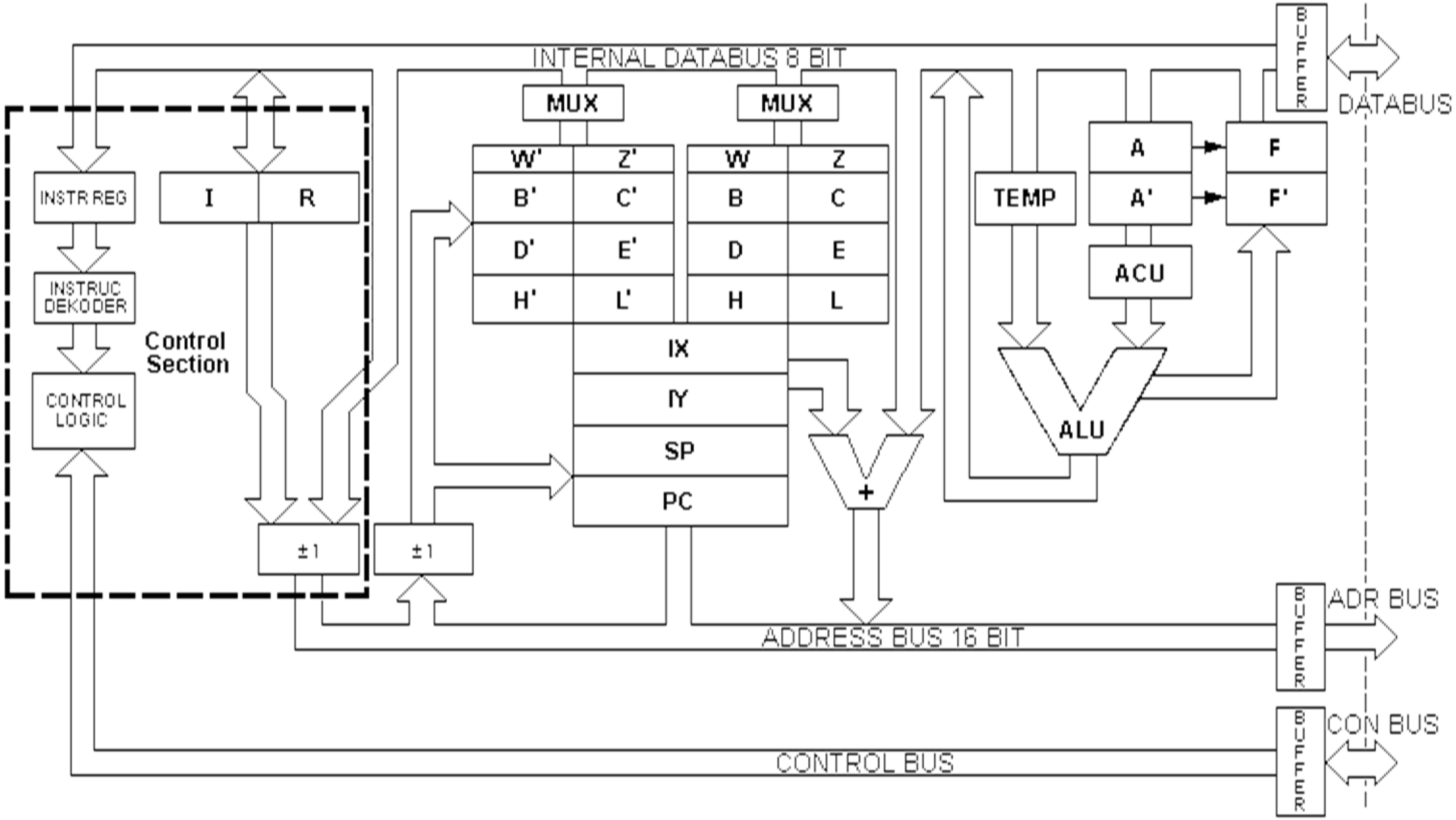
Main Register Set		Alternate Register Set	
Accumulator	Flags	Accumulator	Flags
A	F	A '	F '
B	C	B '	B '
D	E	D '	E '
H	L	H '	L '

General Purpose Registers

Interrupt Vector I	Memory Refresh R
Index Register	IX
Index Register	IY
Stack Pointer	SP
Program Counter	PC

Special Purpose Registers

Figure 2. Z80 CPU Register Configuration



Z80 contains

Control unit: this controls the operation of the CPU through the fetch, decode, execute, read, and write phases.
Buses: data bus, address bus, and control bus.
Register bank: containing data and address registers
ALU: evaluation of logical and arithmetic operations

Z80A Instruction Cycle

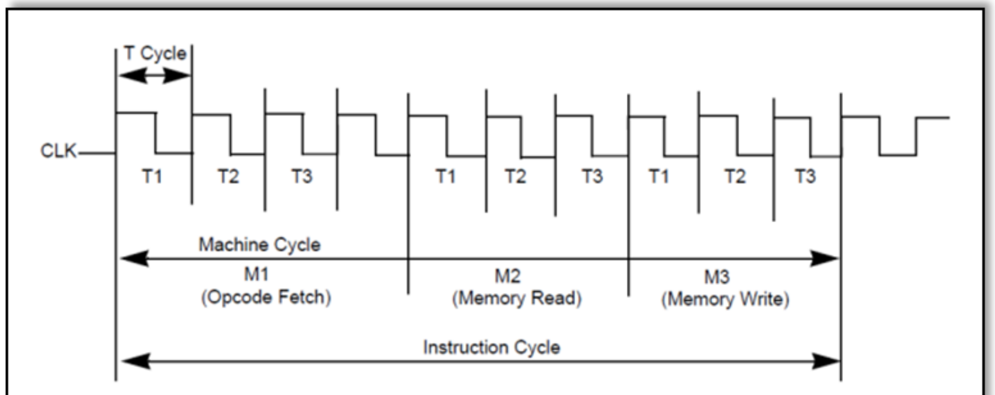
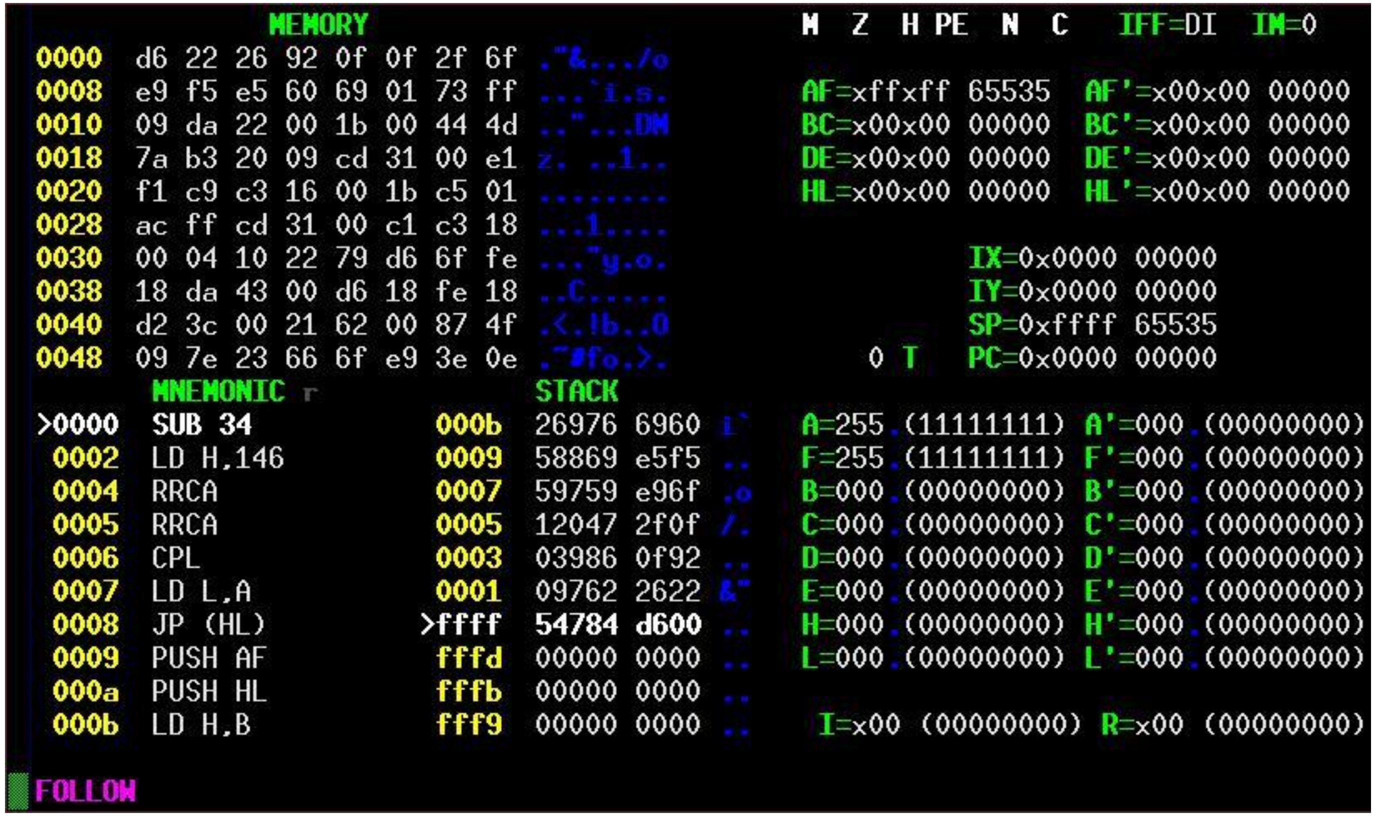


Figure 4. Basic CPU Timing Example



```
                 MEMORY                              M  Z  H PE  N  C   IFF=DI  IM=0
0000  d6 22 26 92 0f 0f 2f 6f  ."&.../o
0008  e9 f5 e5 60 69 01 73 ff  ...`i.s.              AF=xffxff 65535  AF'=x00x00 00000
0010  09 da 22 00 1b 00 44 4d  .."...DM              BC=x00x00 00000  BC'=x00x00 00000
0018  7a b3 20 09 cd 31 00 e1  z. ..1..              DE=x00x00 00000  DE'=x00x00 00000
0020  f1 c9 c3 16 00 1b c5 01  ........              HL=x00x00 00000  HL'=x00x00 00000
0028  ac ff cd 31 00 c1 c3 18  ...1....
0030  00 04 10 22 79 d6 6f fe  ..."y.o.                        IX=0x0000 00000
0038  18 da 43 00 d6 18 fe 18  ..C.....                        IY=0x0000 00000
0040  d2 3c 00 21 62 00 87 4f  .<.!b..O                        SP=0xffff 65535
0048  09 7e 23 66 6f e9 3e 0e  .~#fo.>.                 0 T   PC=0x0000 00000
        MNEMONIC r             STACK
>0000   SUB 34            000b  26976 6960 i`   A=255.(11111111) A'=000.(00000000)
 0002   LD H,146          0009  58869 e5f5 ..   F=255.(11111111) F'=000.(00000000)
 0004   RRCA              0007  59759 e96f .o   B=000.(00000000) B'=000.(00000000)
 0005   RRCA              0005  12047 2f0f /.   C=000.(00000000) C'=000.(00000000)
 0006   CPL               0003  03986 0f92 ..   D=000.(00000000) D'=000.(00000000)
 0007   LD L,A            0001  09762 2622 &"   E=000.(00000000) E'=000.(00000000)
 0008   JP (HL)          >ffff  54784 d600 ..   H=000.(00000000) H'=000.(00000000)
 0009   PUSH AF           fffd  00000 0000 ..   L=000.(00000000) L'=000.(00000000)
 000a   PUSH HL           fffb  00000 0000 ..
 000b   LD H,B            fff9  00000 0000 ..    I=x00 (00000000) R=x00 (00000000)

FOLLOW
```

0x22 = 34 decimal

So d6 opcode for sub

0x92= 146 decimal

So 26 opcode for LD H,

"LD H, = Load H with the value…"

Example instruction cycle (memory-to-memory operation):

- M1=opcode fetch and decode (4 clock cycles)
- M2=memory read (3 clock cycles)
- M3=memory write (3 clock cycles)
- Note: this instruction takes 10 clock cycles

Instruction Fetch

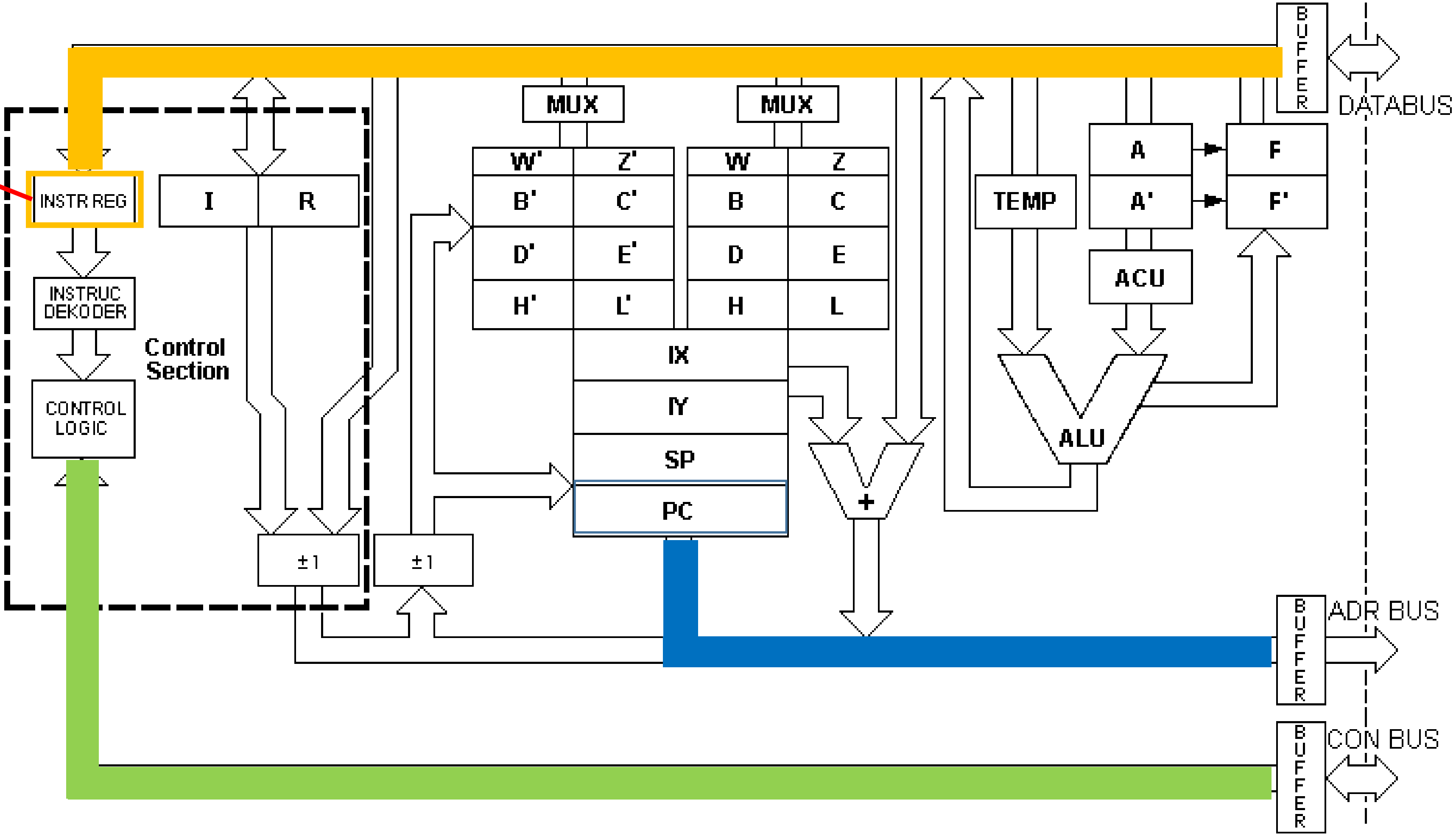
e.g. 26 “LD H,” loaded in here

Fetch causes the first byte to be read this is the opcode.

This sometimes the only byte read from memory.

e.g. nop no operation

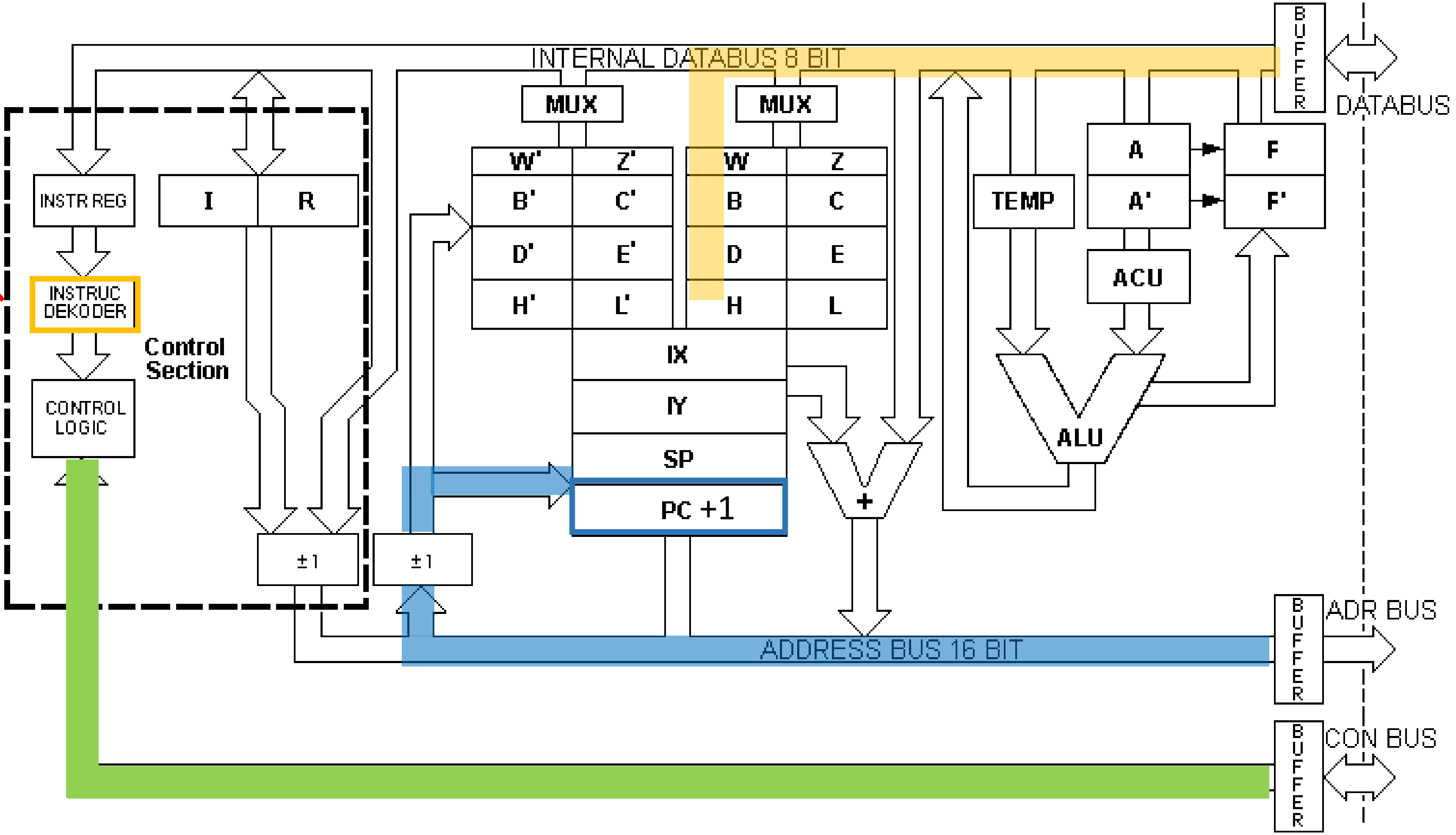
inc a increase a



1. The control section simultaneously:
   - Sends the Program Counter (Instruction Pointer) to the address bus
   - Sends a memory read to the control bus
2. The external memory responds by placing the byte (instruction) at that address on the Data Bus
3. Which is loaded, via the input buffers and the Internal Data Bus, into the Instruction Register

Instruction Decode

e.g. 26 “LD H,” decoded here

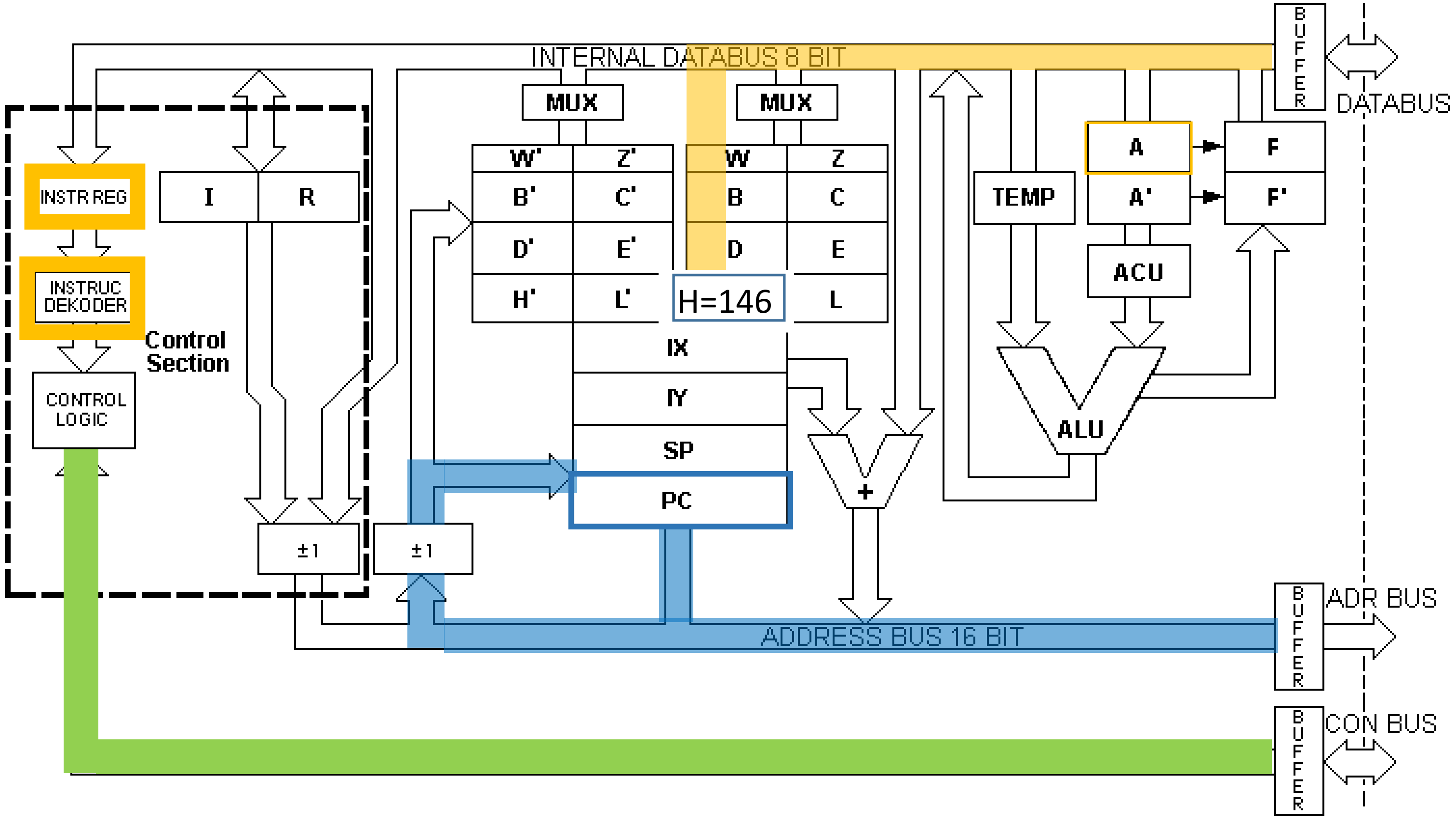


The decoder generates and supplies the control signals.

Get processor ready to read or write data from or to specific registers.

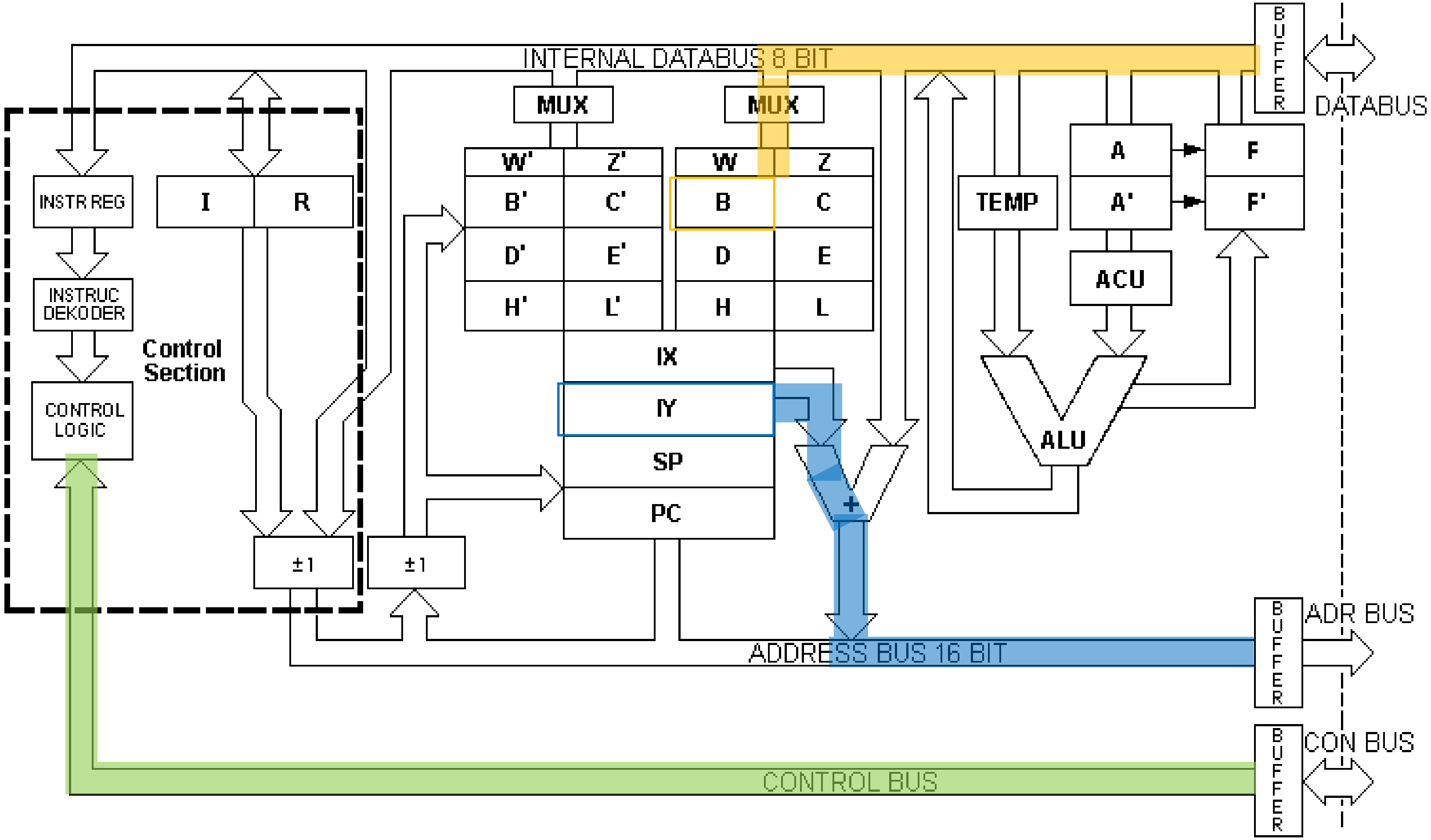
Sets up the ALU.

Read Cycle



- 146- load value from second instruction byte into H
- Control section:
  - Places the value on data bus (146) into H
  - Puts a read request on the control bus for external memory
  - The external memory places the value at this address on the data bus
  - The H register is selected to load a new value from the data bus

Write Cycle



- Write B, (IY)
- Control Section:
  - Places the value in selected address register (IY) onto the internal address bus
  - Selects the B register to put its value on the internal data bus
  - Puts a write request on the control bus for external memory
  - The external memory stores the value from the data bus to this address

Machine cycle

There are many different descriptions of the machine cycle.

Typically they all follow the “fetch”, “decode”, “execute” sequence.

Each step requires one (or more) clock cycles to complete.

On a floating point processor the sequence is “Check if zero”, “Shift significand”, “Add”, “Normalize” (CSAN)