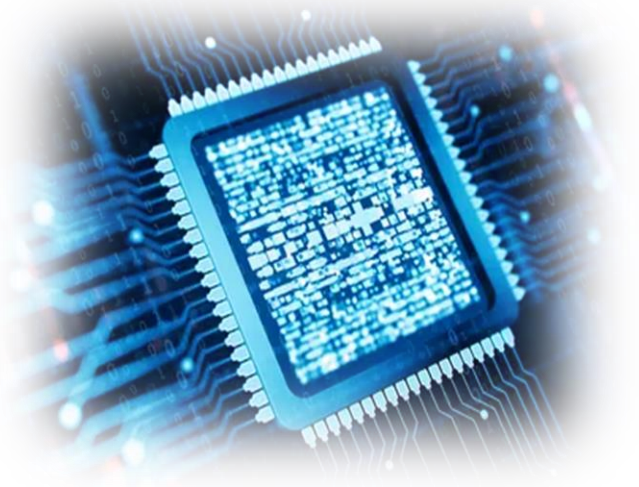


MICROPROCESSOR SYSTEMS

BLG212E

Burak Berk Üstündağ CRN: 11450

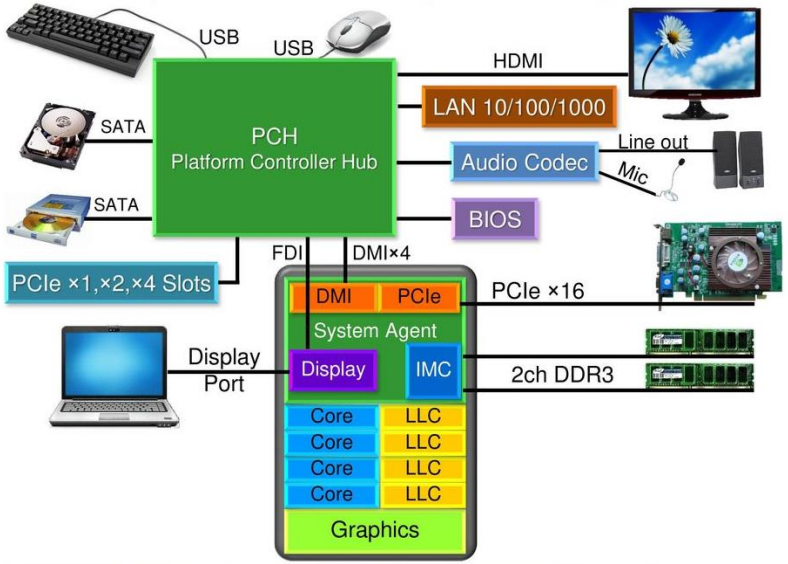
Gökhan İnce / Ayşe Yılmazer CRN: 11446

Faculty of Computer and Informatics Engineering
Istanbul Technical University

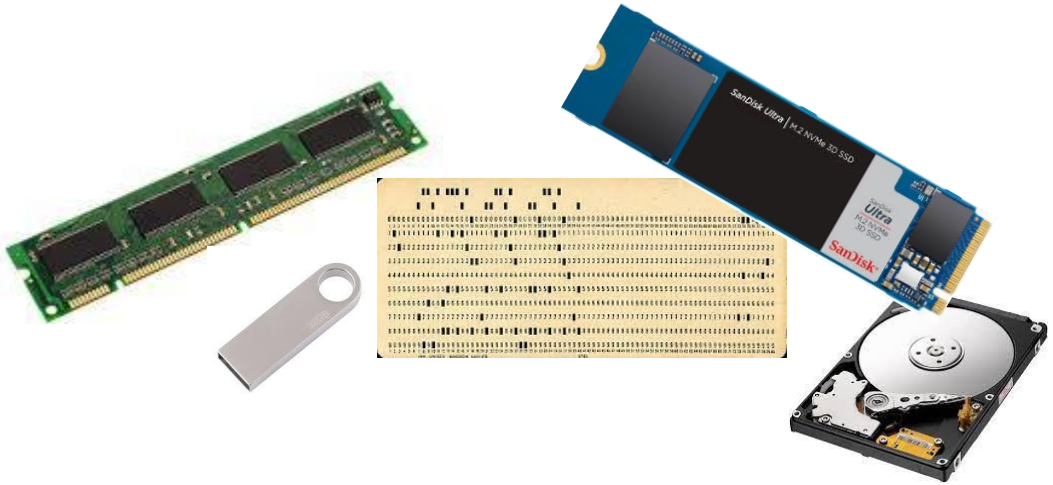
Week 4: Central Processing Units in Computer Structures

Topics

- Computer Structure



- Memory Systems



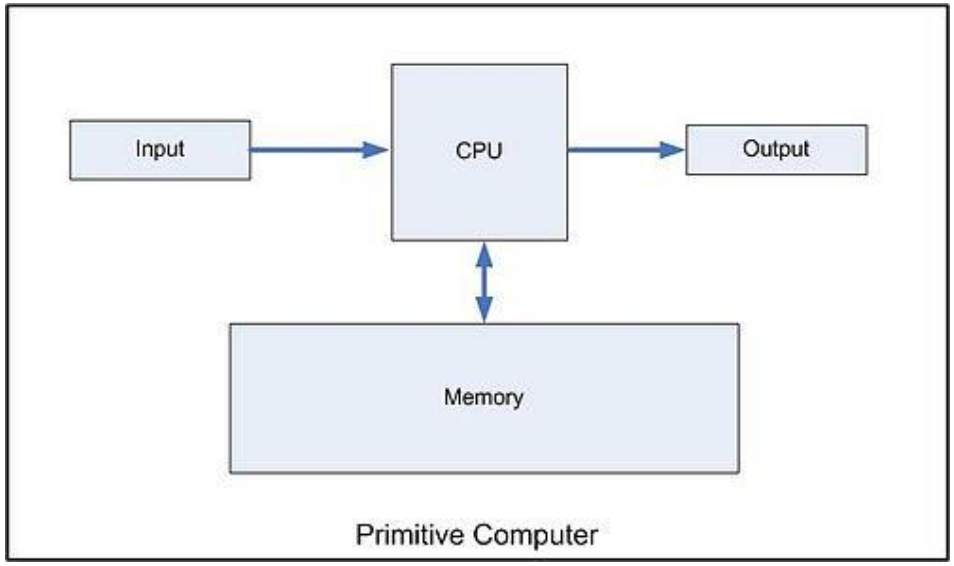
Computer Structure

- A computer is a programmable machine designed to automatically carry out a sequence of arithmetic or logical operations.

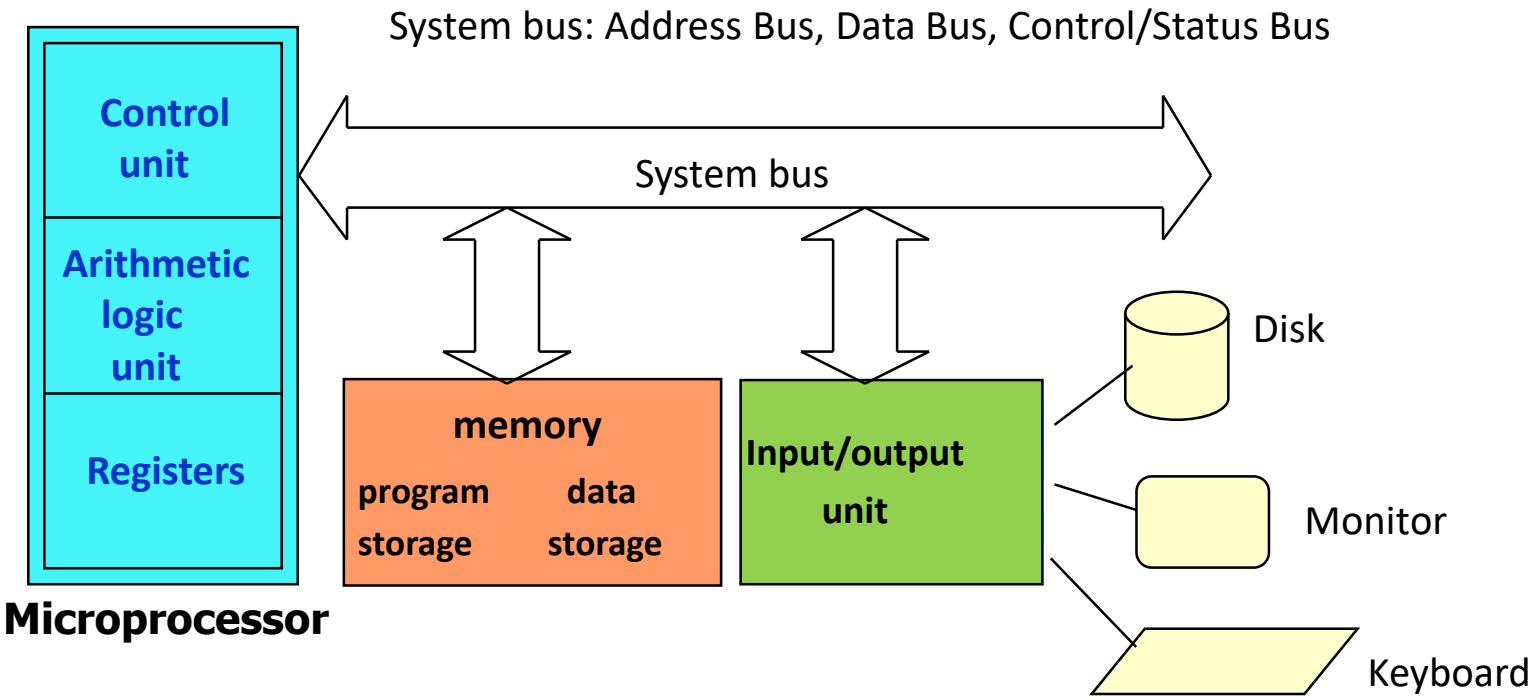
- Fundamental features of a computer
  - Memory capability
  - Computation capability
  - Decision capability
  - Input/Output capability

What is a computer ?

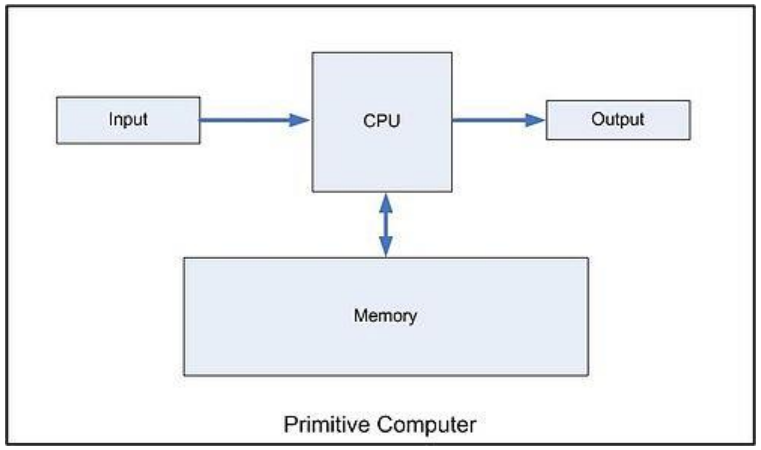
- It is programmable – where does the code reside ?
  - Memory
- Executes or processes the instructions of the program or code
  - Arithmetic and logic processor
- How does the code or data get in to the computer?
- How does it output the result?
  - Input/Output device(s)

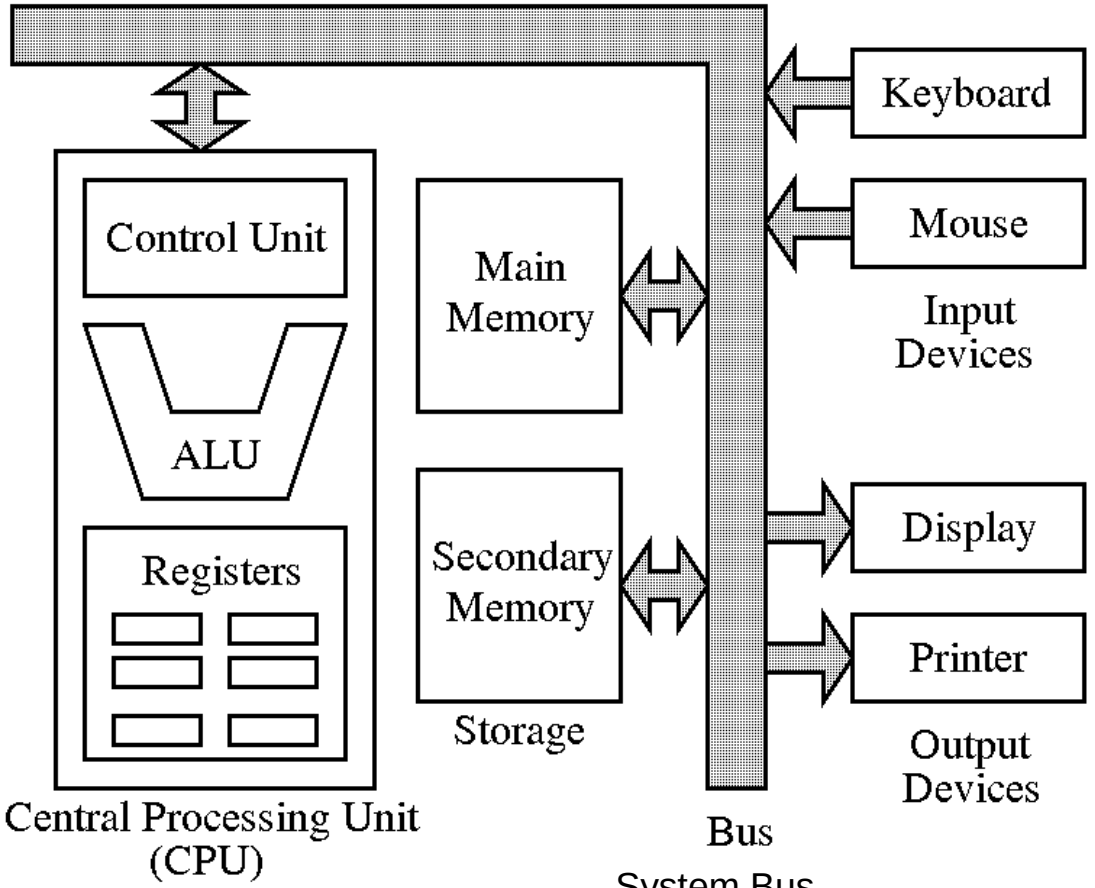


Computer Organization



What is a computer ?





System Bus
(Common or Separate):
Address bus,
Data bus,
Control bus

Memory

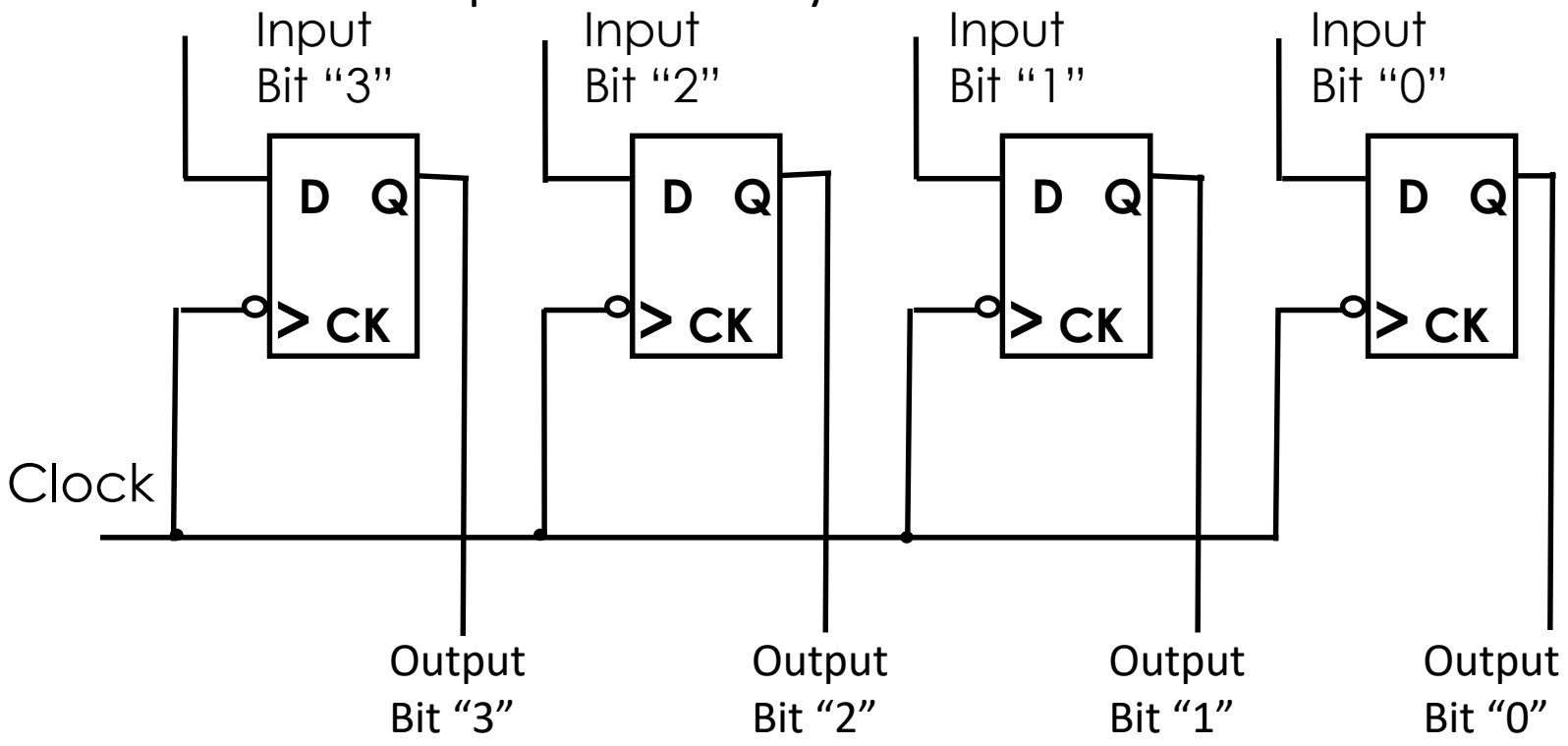
- Memory stores information such as **instructions** and **data** in binary format (0's and 1's). It provides this information to the microprocessor whenever it is needed.

- Usually, there is a memory "sub-system" in a microprocessor-based system. This sub-system includes:
  - The registers inside the microprocessor
  - Read Only Memory (ROM)
    - used to store information that does not change.
  - Random Access Memory (RAM) (also known as Read/Write Memory)
    - used to store information supplied by the user such as programs and data.

Registers

- A **flip flop** is a 1-bit memory device.
- A **register** is a collection of flip flops that are clocked as a unit.
- Combinational vs. sequential circuitry

D	Q(t+1)
0	0
1	1



Registers

- Register is a group of n-flip-flops capable of storing n-bits of information
- Registers are accessed **very fast** within the CPU
- Different CPUs have different set and size of registers
- **General-purpose** registers are used to store data during the CPU execution process
- **Special-purpose** registers control or monitor various aspects of the microprocessor's function (MAR, IR...).

CPU Structure

- **Memory Address Register** (MAR) either stores the memory address from which data will be fetched to the CPU or the address to which data will be sent and stored.

- **The Memory Data Register** (MDR) contains the data to be stored in the computer storage (e.g. RAM), or the data after a fetch from the computer storage.

- **Accumulator** (ACC) stores intermediate arithmetic and logic results.

Arithmetic and Logic Unit (ALU)

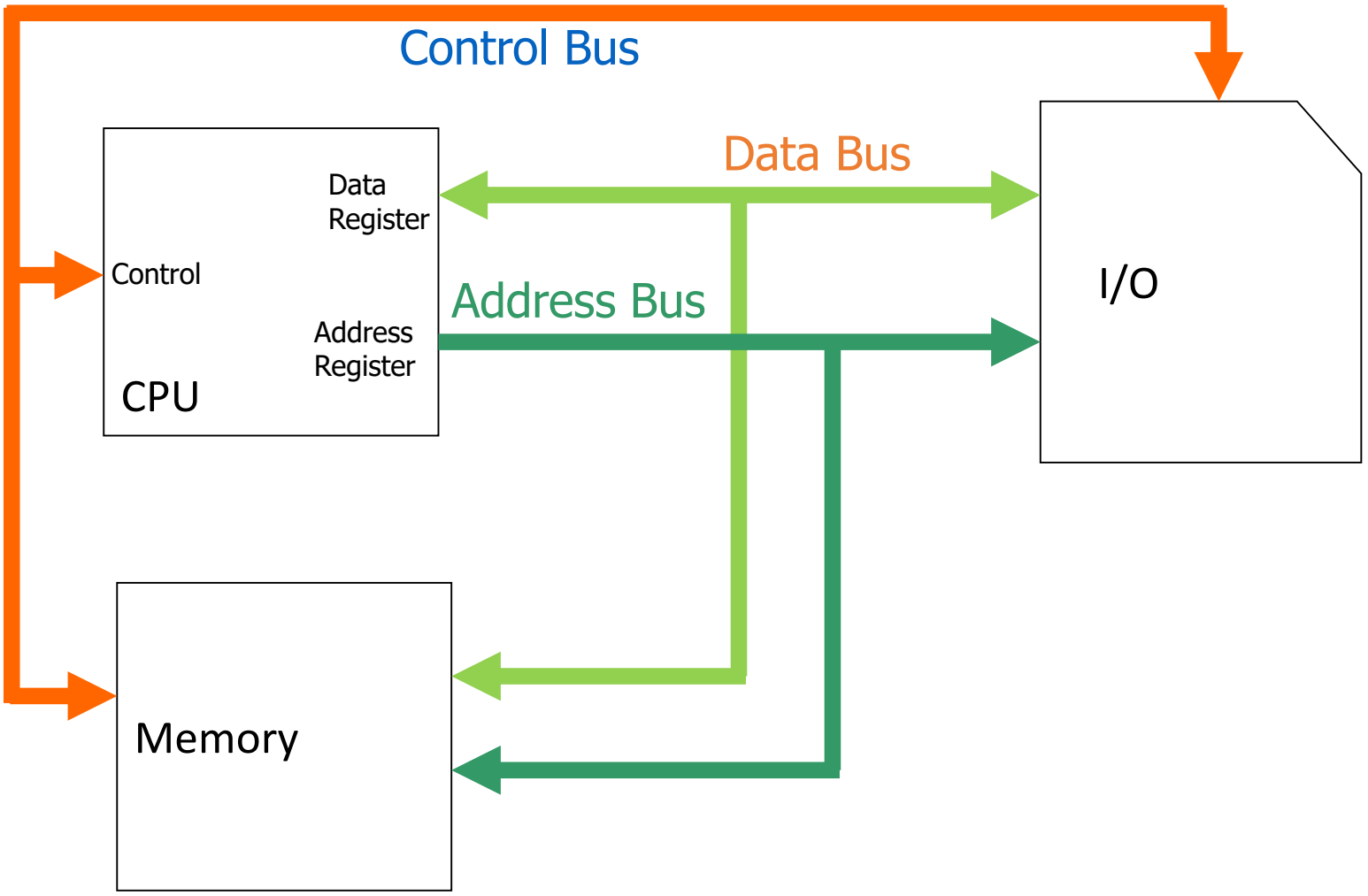
- **Arithmetic and Logic Unit** (ALU) performs arithmetic and logical operations
- The ALU receives data from main memory and/or the register file, performs a computation, and, if necessary, writes the result back to main memory or registers.

Control Unit

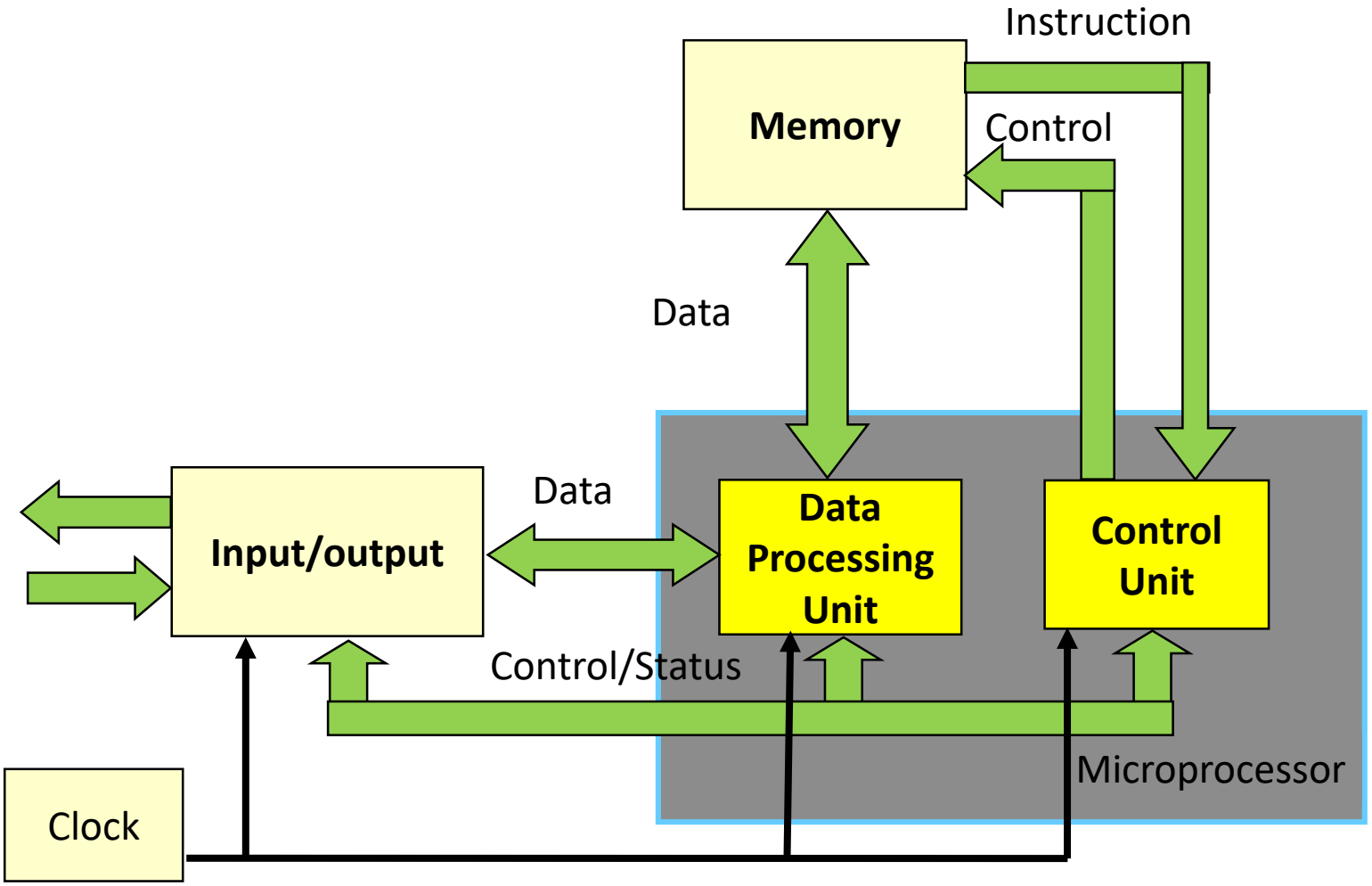
- **Control Unit (CU)** manages the computer operation within and outside the CPU

  - Sequencing and execution cycles of commands
  - Register operations
  - Bus regulation
  - Interrupt handling
  - System status
  - Memory management

Basic Computer Connections

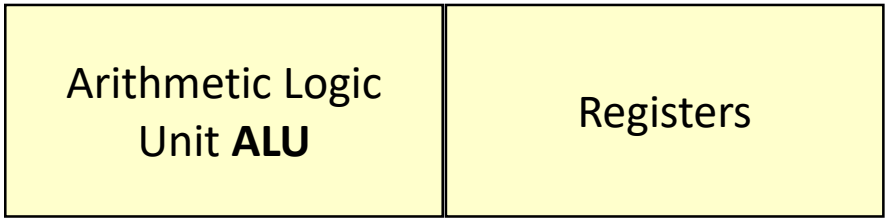


Basic Computer Architecture



Stored Program Procedure

- **The Clock:** The stored program processor is a large synchronous sequential network (logic circuit) that requires a clock signal to **synchronize** all its elements.
  - The signal from the clock is a **periodic pulse waveform**.
- **The Data Processing Unit:** The principal part of the data processing unit is a combinational logic circuit, called the ALU, that manipulates binary numbers.
  - Practical **ALUs** can perform dozens of different operations, but only one at time. Usually ALUs can add, subtract, complement bits, shift binary values and so on.
  - **Registers** hold the data numbers operated or produced.

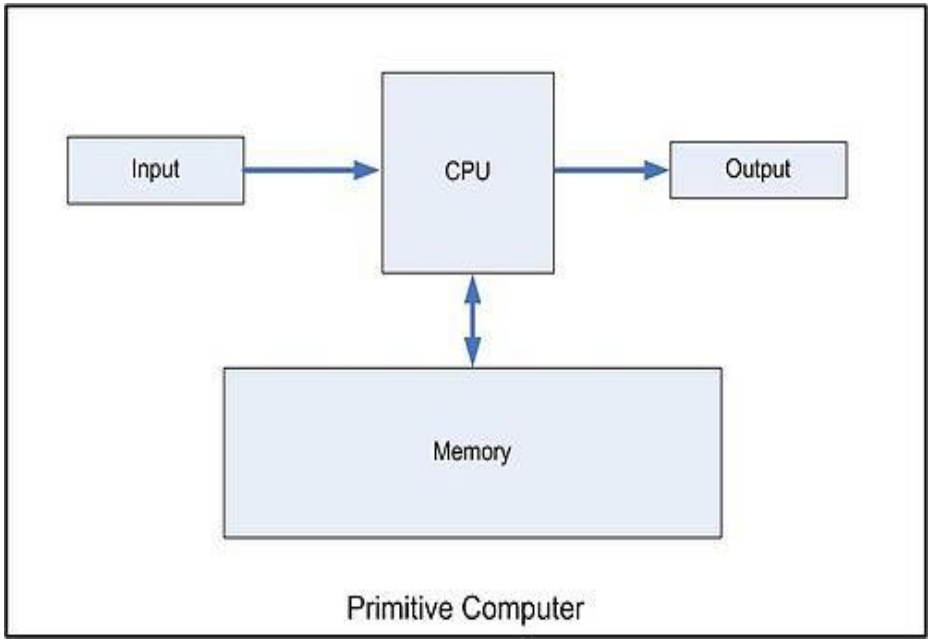


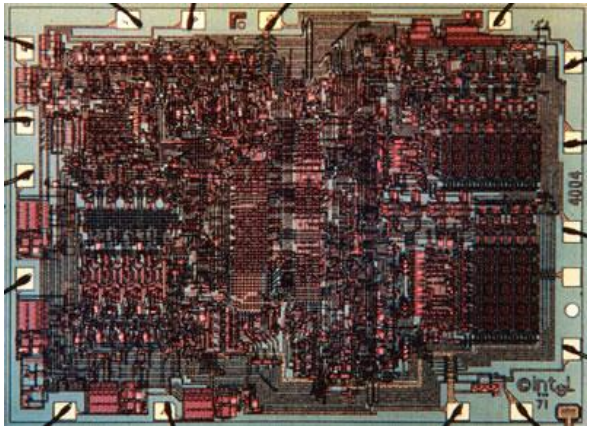
Stored Program Procedure

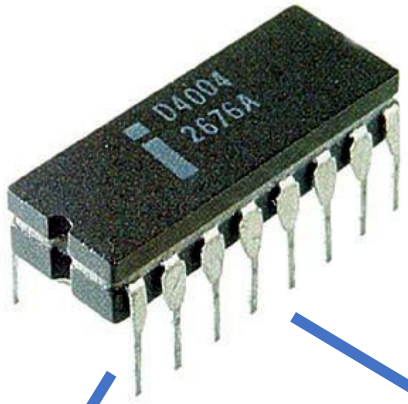
- **The control unit** is a synchronous sequential logic circuit that sends **control signals** to the data processing unit, memory and other parts of the system.
  - The signals from the control unit tells the data processing unit to manipulate data according to the algorithm built into the sequential logic circuit.
  - The control unit is **instruction controlled**; therefore it can do more than one algorithm based on its design. Typical control units recognize several hundred different instruction codes.
- **The Memory:** The memory holds instruction code numbers and data numbers
- **The I/O Unit:** The input/output unit includes any hardware that allows data transfer between the CPU and the real world.

Central Processing Unit (CPU)

- CPU is the fundamental execution/processing unit of the computer
- CPU consists of ALU, Control Unit, and Registers
- CPU is characterized by:
  - Clock frequency
  - Speed
  - Data bus width
  - Instruction set
  - Addressing capability
  - Addressing capacity







Intel 4004

- in 1971, commercially available single-chip microprocessor
- 0.092MIPS at 0.740 MHz
- 12 bit address bus
- 4 bit data bus

TMS1000, 1974
First Microcontroller
4Bit, 1K ROM, 64B RAM
Texas Instruments
Gary Boone and Michael J. Cochran

8085: 8 bits CPU,
1976
3.5MHz, 16Bit
Address Bus, DMA

8x48: 8 bits
microcontroller, 1976
1K EPROM, 64/128 SRAM

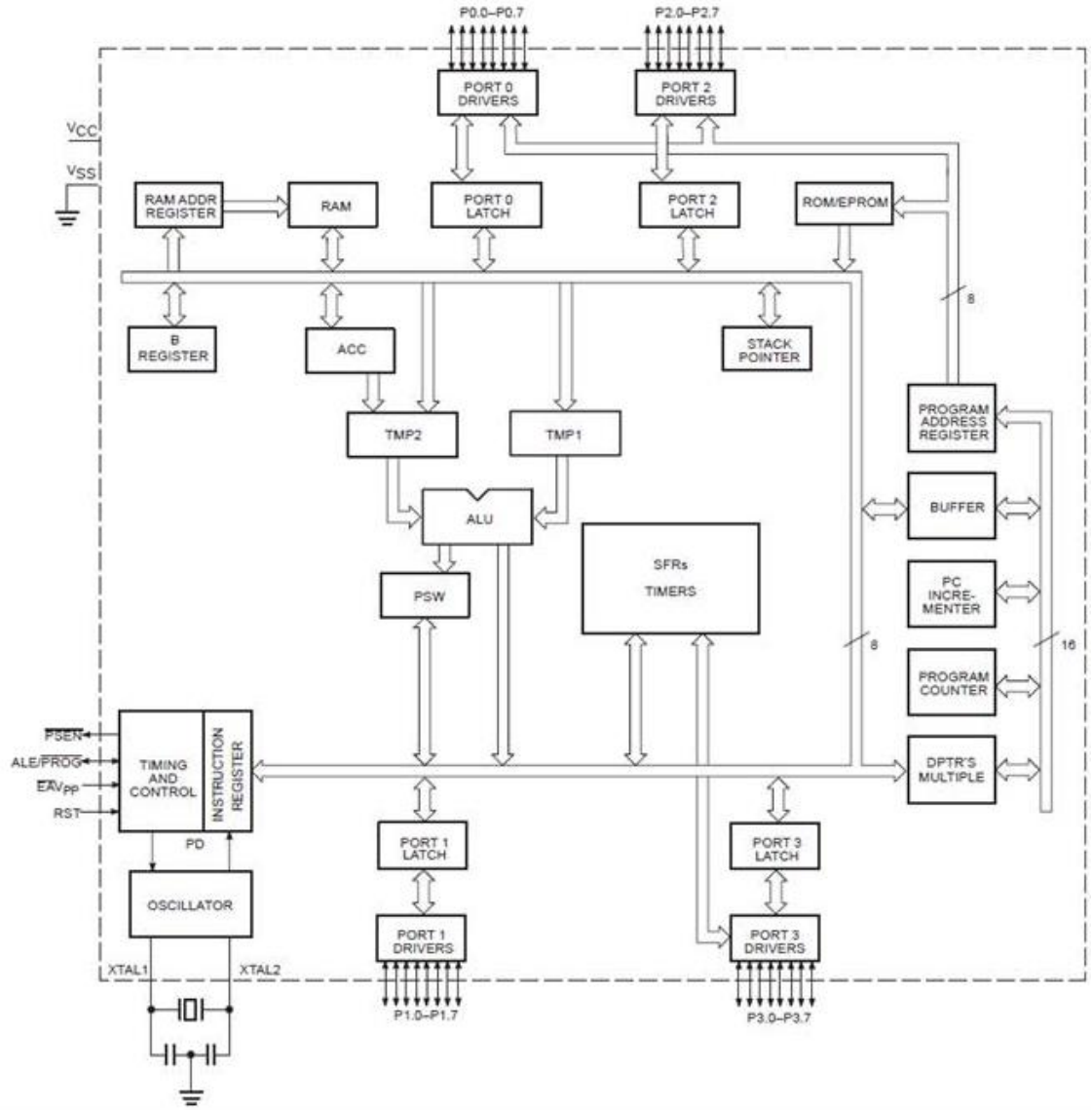
8086 CPU: 1978, 16bits Data Bus 5MHz/10MHz
8088 CPU: 1979, 8Bits (First IBM PC in 1981)
80286 CPU: 1982, 16Bit data /20Bit Address 6..25MHz
80386 CPU: 1985, 32Bit, 275.000transistors, 20..25MHz
80486 CPU: 1989, 50MHz, 32Bit data / 32Bit Address, 50MIPS
Pentium CPU: 1993, 60MHz…133MHz L1,L2 Cache
…..
I7 CPU: 2008, 2.8…3.5GHz, L1,L2,L3 Cahce
I9 CPU: 2017, 3.3…4.5GHz, L1,L2,L3 Cahce 412,090MIPS at 4.7 GHz (I9-9900)

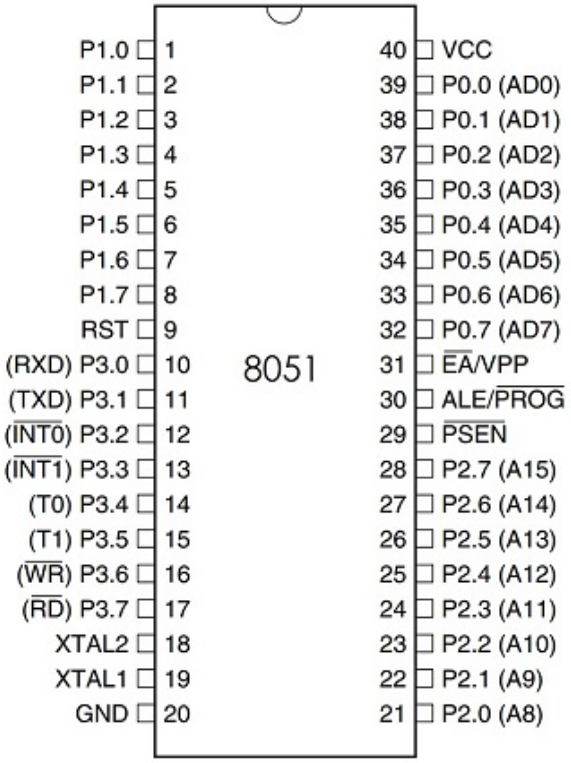
8x51 µC: 1978, 8bits, 8K / 256B RAM
8x96 µC: 1984, 16bits & Motorola 68HC11-8Bits
8x196 µC: 1988
…….

CPU Elements

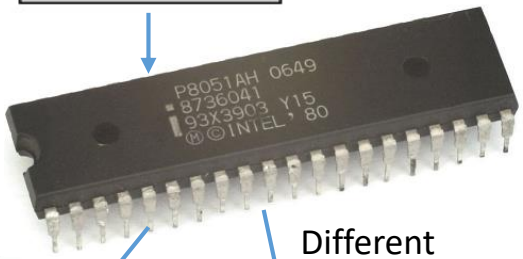
- Program Counter (PC)
- Instruction Register (IR)
- Instruction Decoder
- Arithmetic and Logic Unit (ALU)
- General Purpose Registers
- Special Purpose Registers (SP, BP, IX, CCR, etc.)
- Control Unit (CU)

8x51 family is one of the oldest and widest manufactured industry standard single-chip µC

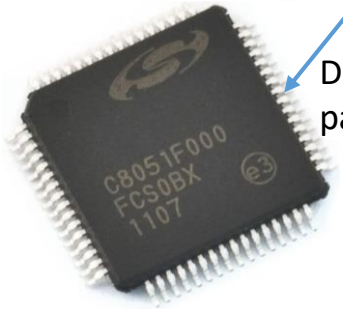




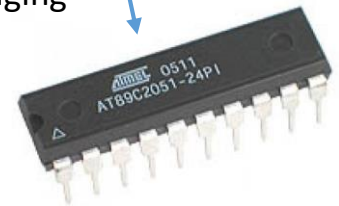
DIP



Different packaging

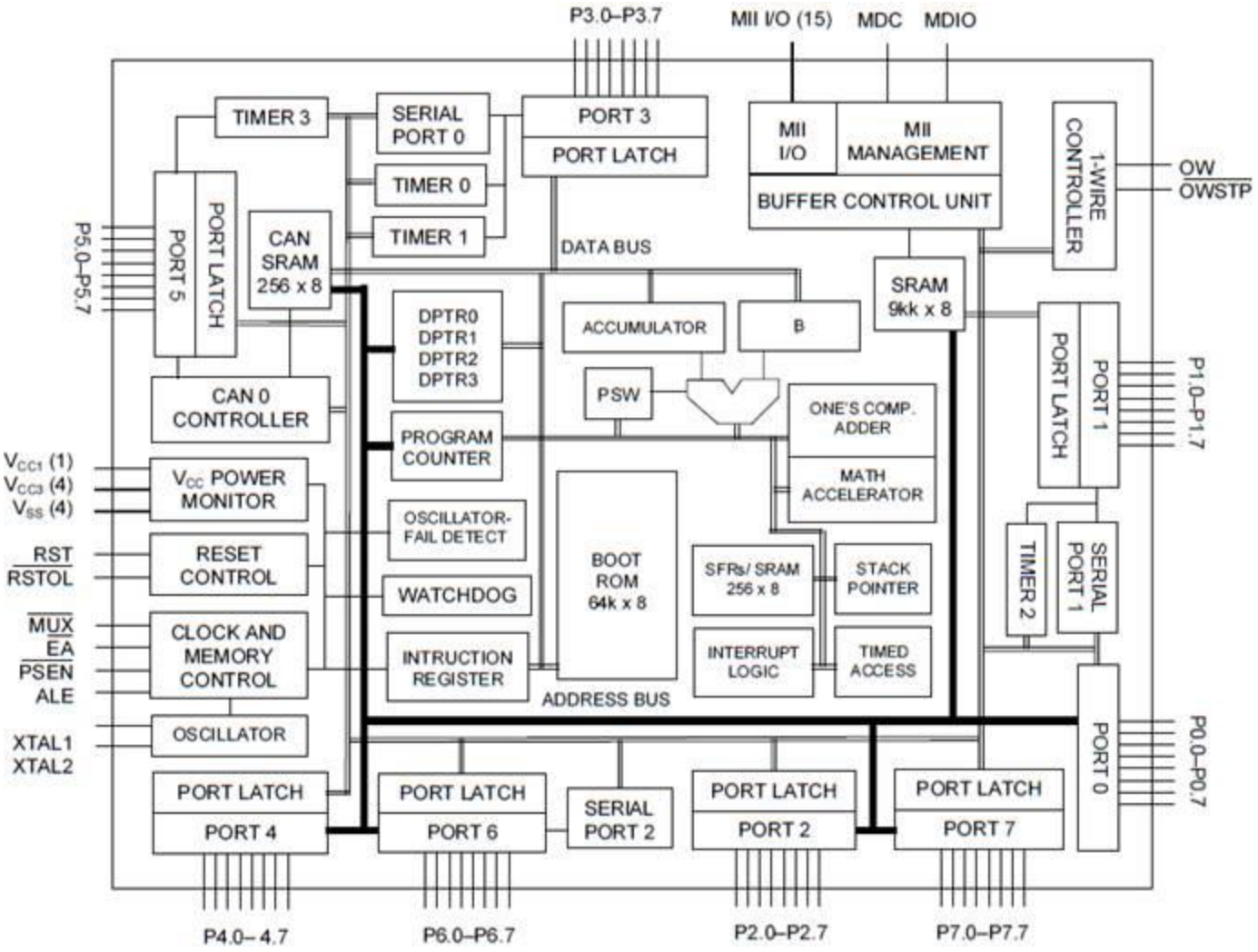


Different I/O & memory configurations



Philips 80C51/87C51/80C52/87C52

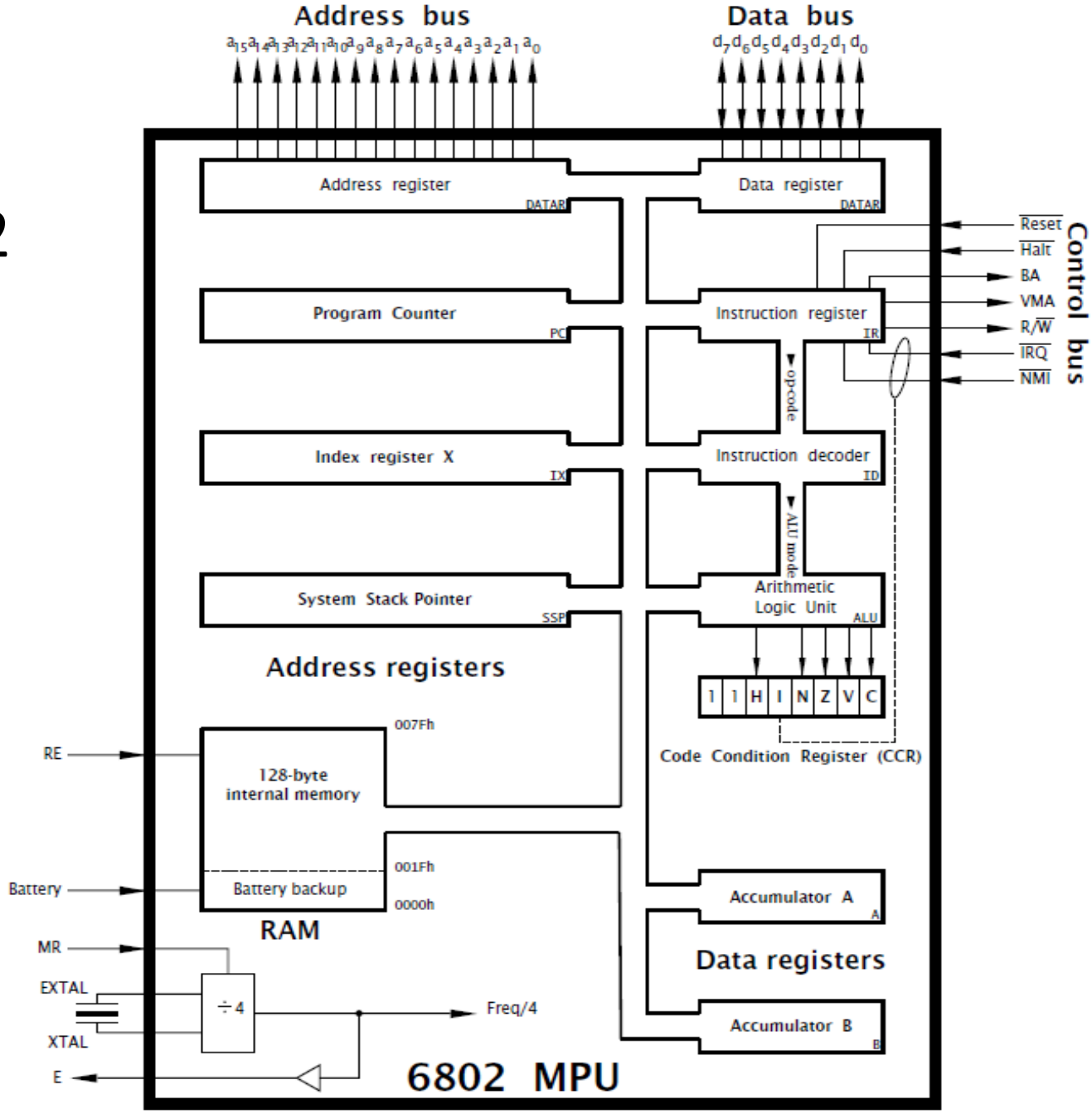
Several derivatives are manufactured based on the same CPU and the internal bus structure



DALLAS MAXIM DS80C400
(8x51 Derivative as Network Controller)

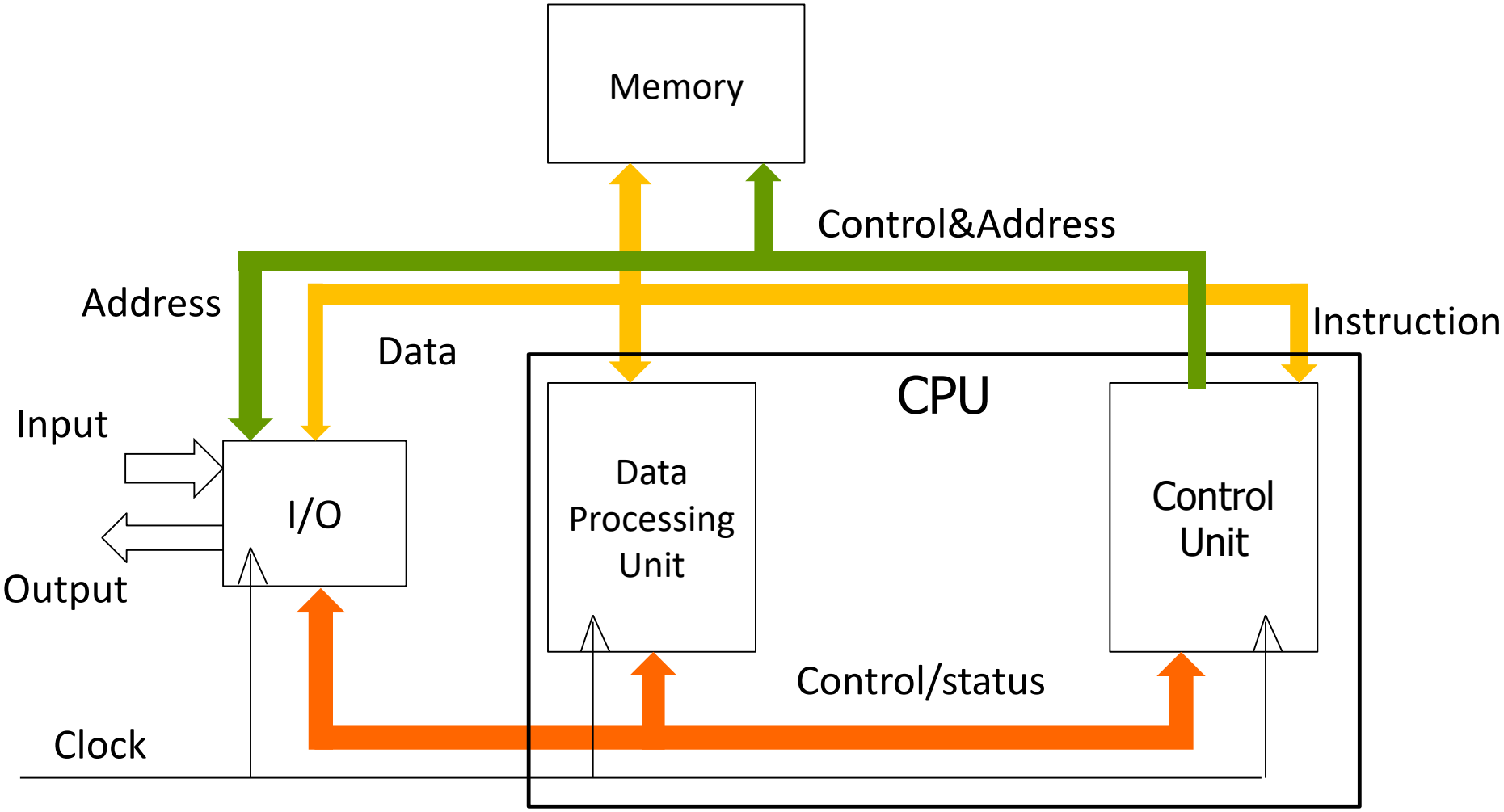
Internal Structure of CPU

Example: Motorola 6802



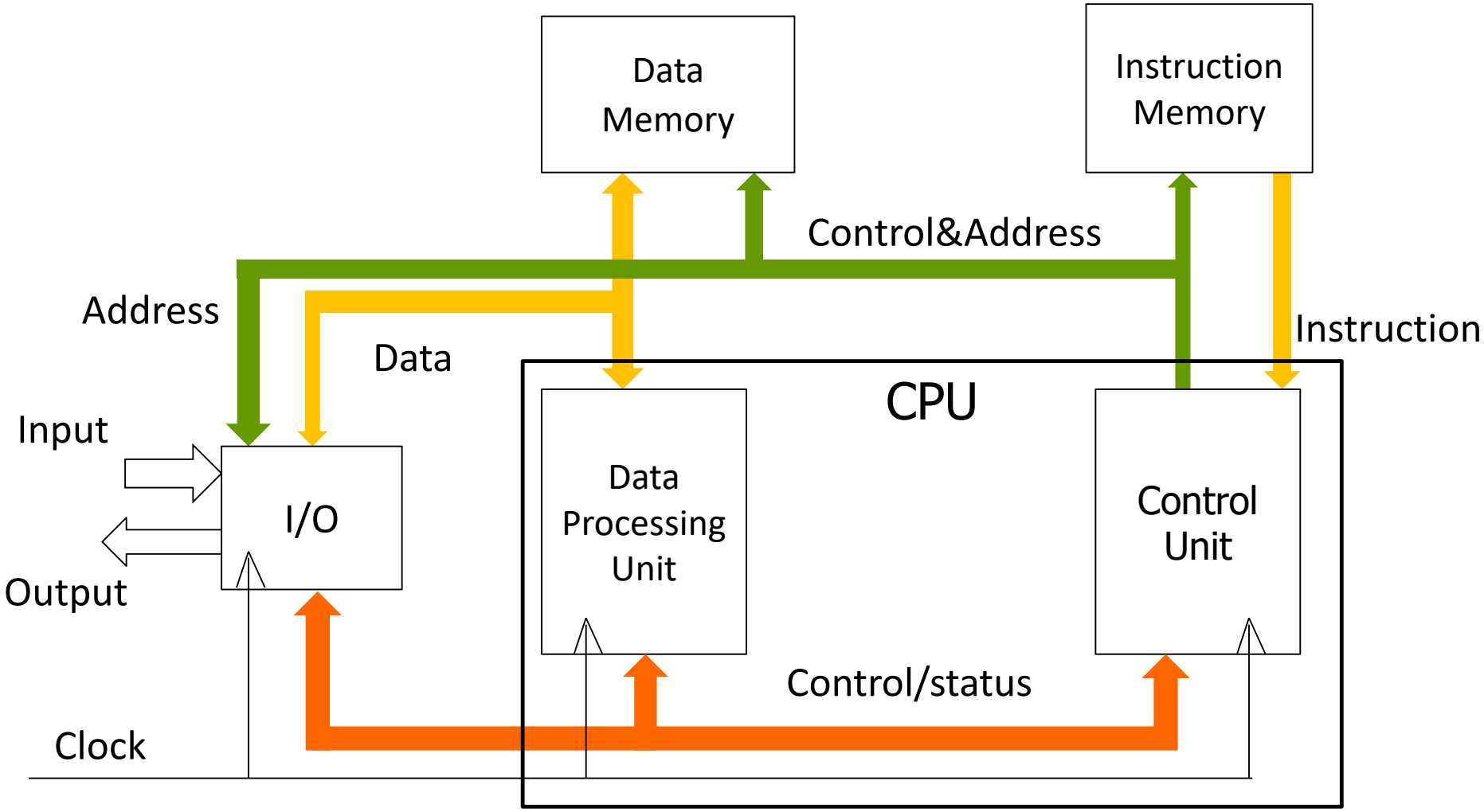
Von Neumann Architecture

Instruction and data are in the same memory



Harvard Architecture

There are two memories: Instruction and data



Operation of Computer

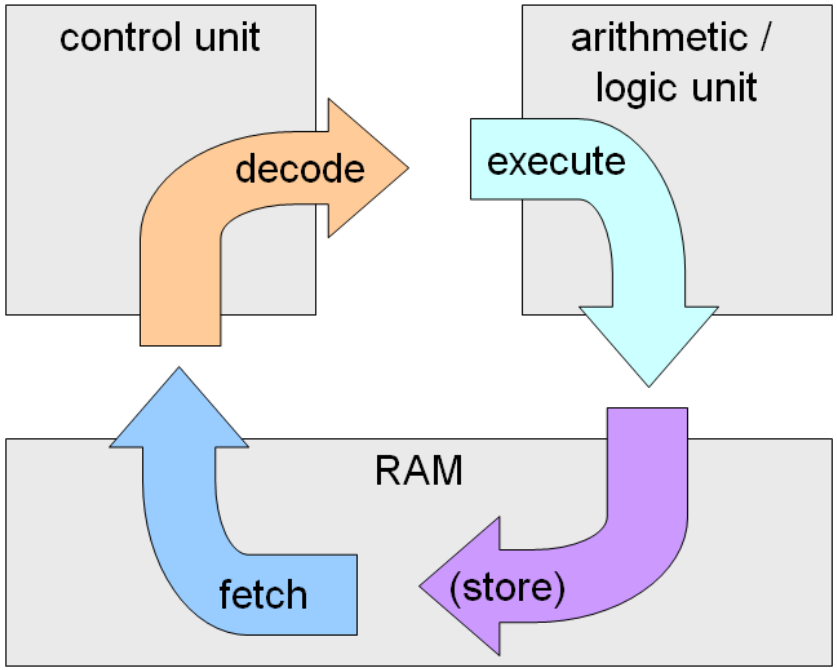
- Example: The X and Y numbers will be written on memory addresses $3000 and $3001. The sum of X and Y will be stored on address $3002

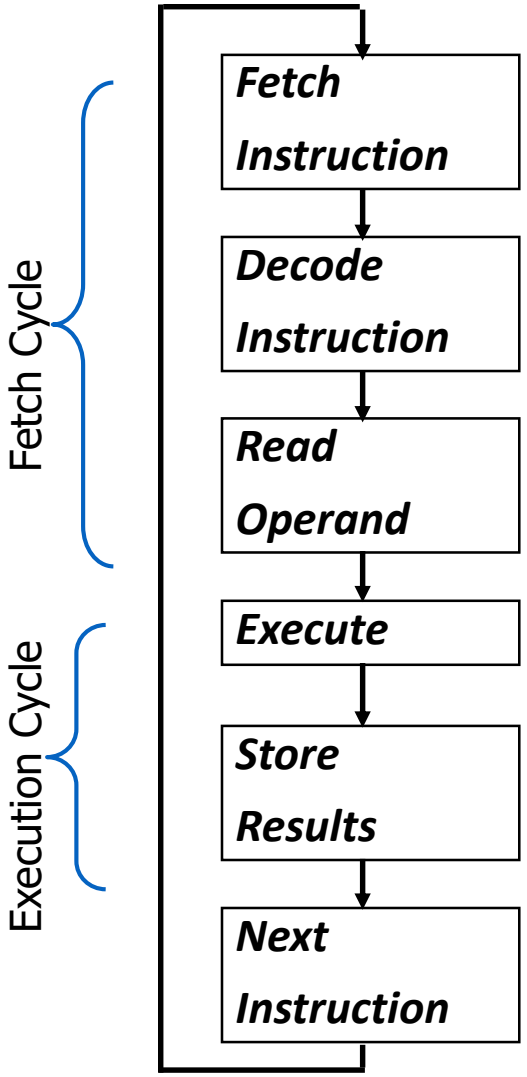
Memory Address	Instruction
$1001	Load X to ACC (accumulator)
$1002	Store contents of ACC to address $3000
$1003	Load Y to ACC
$1004	Store contents of ACC to address $3001
$1005	Add contents of $3000 to ACC
$1006	Store contents of ACC to address $3002

$3000	X	Results after execution
$3001	Y	
$3002	X+Y	

Instruction Cycle

Fetch-Decode-Execute

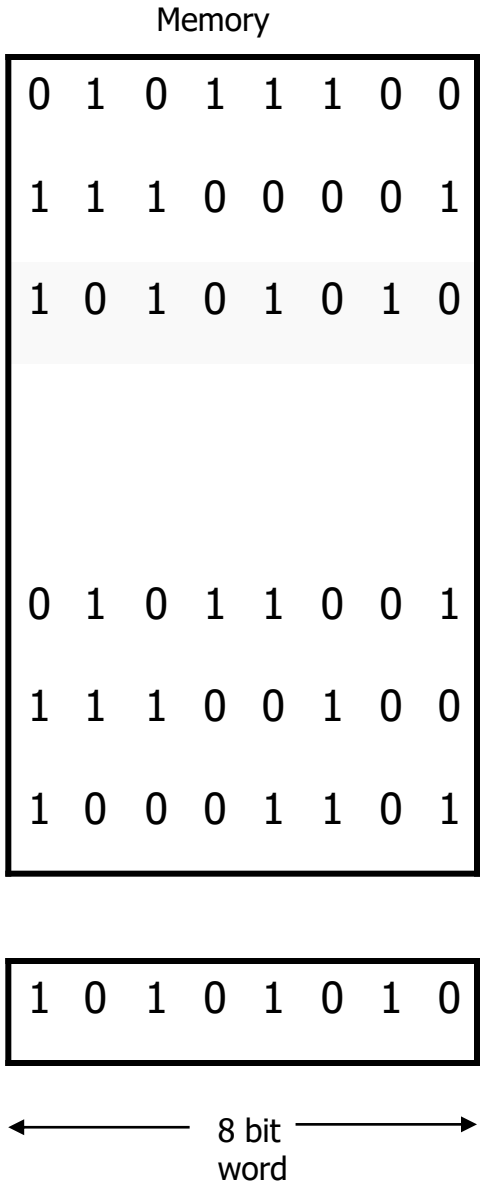




- Two-cycle process because both instructions and data are in memory
- Fetch
  - Decode or find instruction, load from memory into register and signal ALU
- Execute
  - Performs operation that instruction requires
  - Move/transform data

Instruction Formats

- Instructions are stored in the memory
- They are executed in a sequence
- They instruct the computer what to do
- The computer fetches the next instruction and decodes it (See Instruction Cycle).
- Each CPU has different set of instructions
- Each CPU has different template of instructions



Instruction Templates

- 3+1 Address instruction template (Example: ADD X,Y,Z,N)

Opcode	1.Operand address	2.Operand address	Result address	Address of next instruction

- 3 Address instruction template (Example: ADD X,Y,Z)

Opcode	1.Operand address	2.Operand address	Result address

- 2 Address instruction template (Example: ADD X,Y)

Opcode	1.Operand address	2.Operand address

- 1 Address instruction template (Example: ADD A,X)

Opcode	Register	Operand address

- 0 Address instruction template (Example: PSH A, PUL A)

Opcode	Register

Long instructions can take up multiple words in memory

Example: 8-bit words in memory, instruction template can be 40bits, takes up 5 words in memory

0, 1, and 2 address instructions are commonly used in modern computers

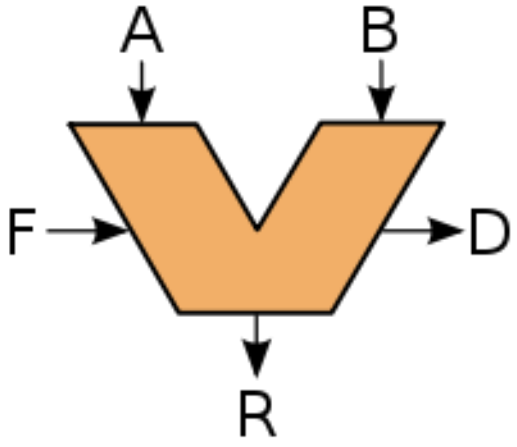
Registers in the Fetch Unit

- **Program Counter:** holds the memory location of the next instruction.
- **Instruction Register:** holds the current instruction being executed

Instruction Decoder

- It decodes the instructions and generates the control signals

Arithmetic Logic Unit (ALU)



- ALU performs all arithmetic and logic operations in a microprocessor
- ALU has two inputs (A, B) for the operands and one input for a control signal that selects the operation
- Operation and Shift control bits determine, which type of operation to perform (F)
- Output is the result of operation (R) and status information (D)
- Status information is used to indicate cases
  - Zero: if all result lines have value 0
  - Overflow: integer overflow of add and subtract functions
    - For unsigned integers, it does not provide any useful information

Registers

- A register is a storage location in the CPU
- It is used to hold data or a memory address during the execution of an instruction
- Because the register file is small and close to the ALU, accessing data in registers is much faster than accessing data in memory outside the CPU
- The register file makes program execution more efficient
- The number of registers varies from computer to computer

Condition Code Register or Flag Register

- Depending on the outcomes of Arithmetic or Logical operations, we can *branch* and *jump*
- The eight-bit Condition Code Register (CCR) provides a status report on the ALU's **activity**
  - **C**arry/Borrow
  - **H**alf carry from bit 3 to bit 4
  - o**V**erflow
- CCR also provides a status report **after loading** ACC
  - **Z**ero
  - **N**egative

			V	Z	N	H	C

Condition Code Register (CCR)

- They flag certain conditions resulting from the ALU outcomes
- Example:

  A= 01001000 B= 01111001

  A+B:

  A 01001000

  B +01111001

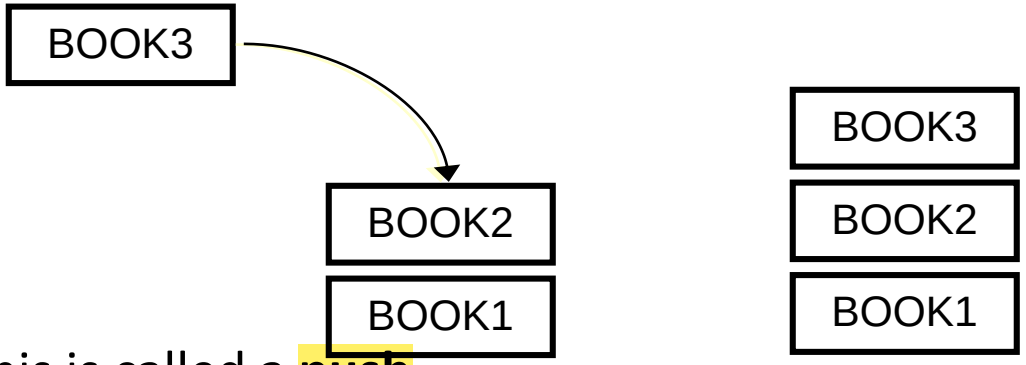
  11000001 V=1 Z=0 N=1 H=1 C=0

- Depending on the outcomes of Arithmetic or Logical operations, we can branch and jump

The Stack

A stack is a last-in-first-out data structure

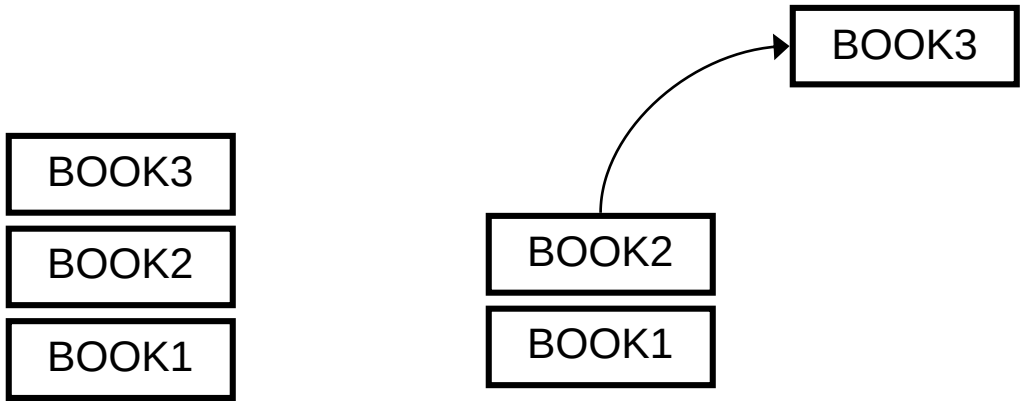
- A stack of a computer works just like a real stack, e.g., of books. If you have a stack of books, you can put another book on top:



- This is called a **push**
- All that happens is the stack gets one book deeper, and the last book you added is on top

The Stack

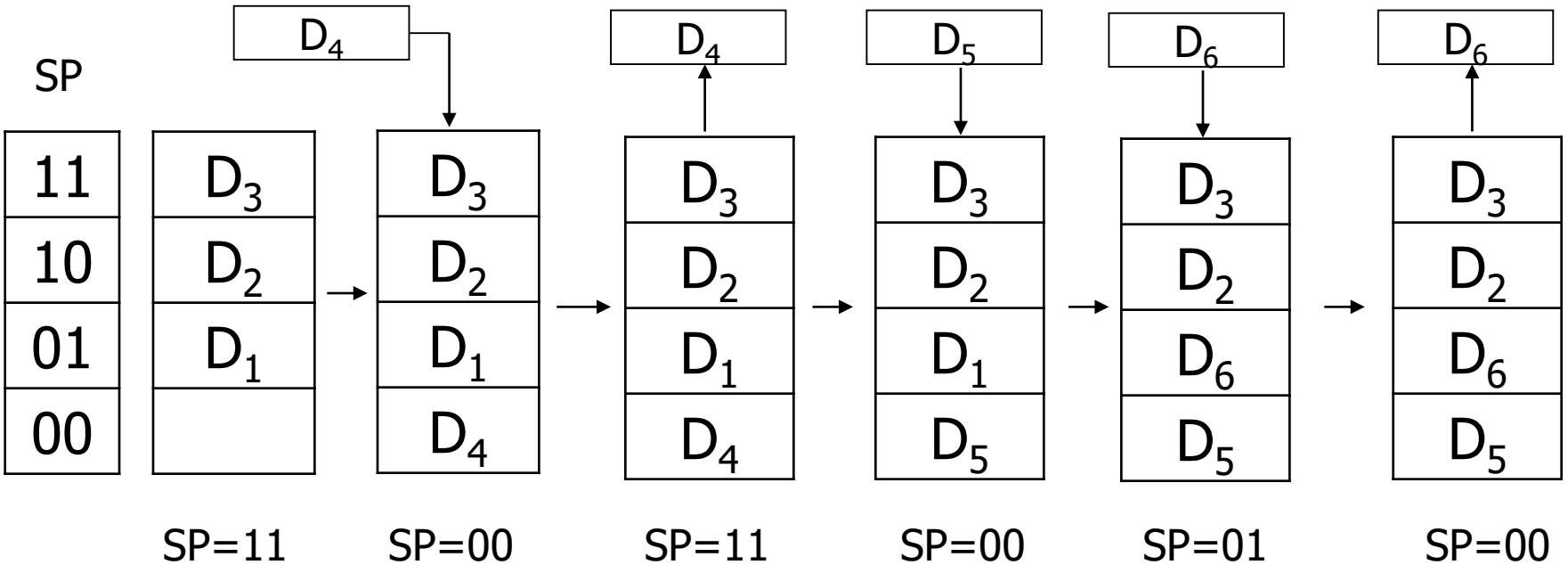
- You can also take a book off the top of the stack:



- This is called a **pop**.
- The stack gets one book shorter, and the book you get from the top is the one you added, or pushed, most recently
- Because a pop gives you back the item you most recently pushed, a stack is called a **last-in-first-out**, or **LIFO**, structure

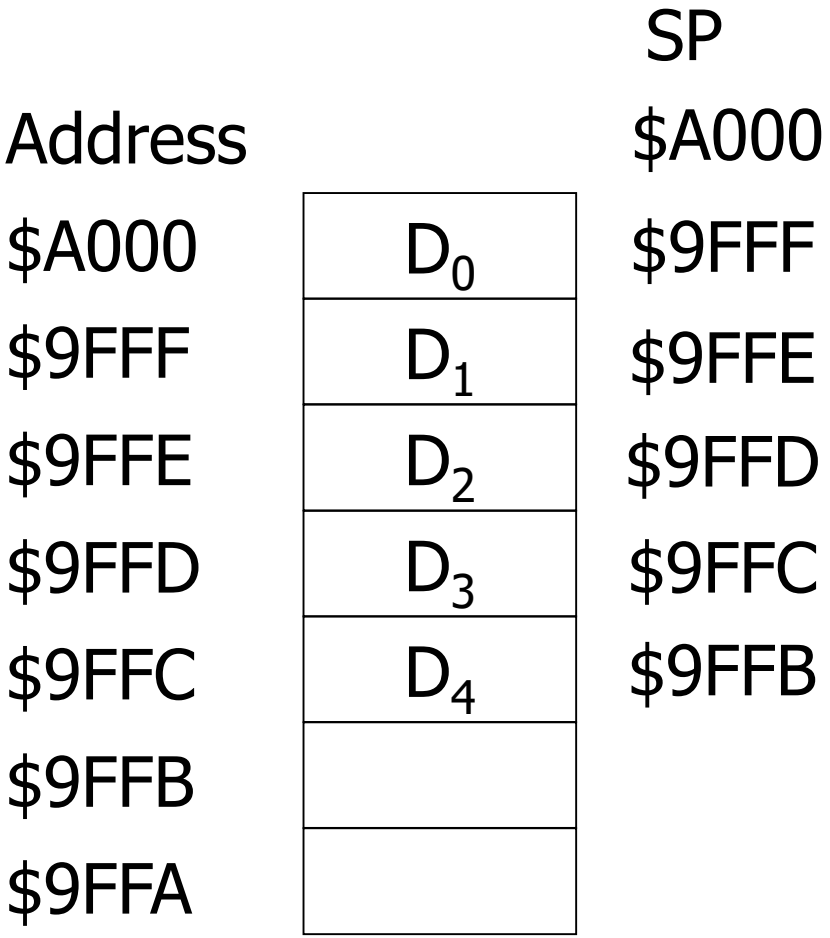
The Stack

A stack is a last-in-first-out data structure.



Stack Pointer

- The stack is a way of using the memory.
- All that's needed is some **unused memory and an index register, called the Stack Pointer** (SP), that always points to the next available (empty) location above the current top of the stack
- The stack grows toward lower addresses



Control Unit

- The control unit is a **synchronous sequential** logic circuit that sends control signals to the data processing unit, memory and other parts of the system
- The signals from the control unit tells the data processing unit to manipulate data according to the algorithm built into the sequential logic circuit
- The control unit is **instruction controlled**; therefore it can do more than one algorithm based on its design (programmable)
- Typical control units recognize several hundred different instruction codes

System Clock

- In order to regulate when the control unit issues its control signals, computers use a system clock

- System clock generates **regular pulses** to synchronize all system events and determine the speed at which processing can occur

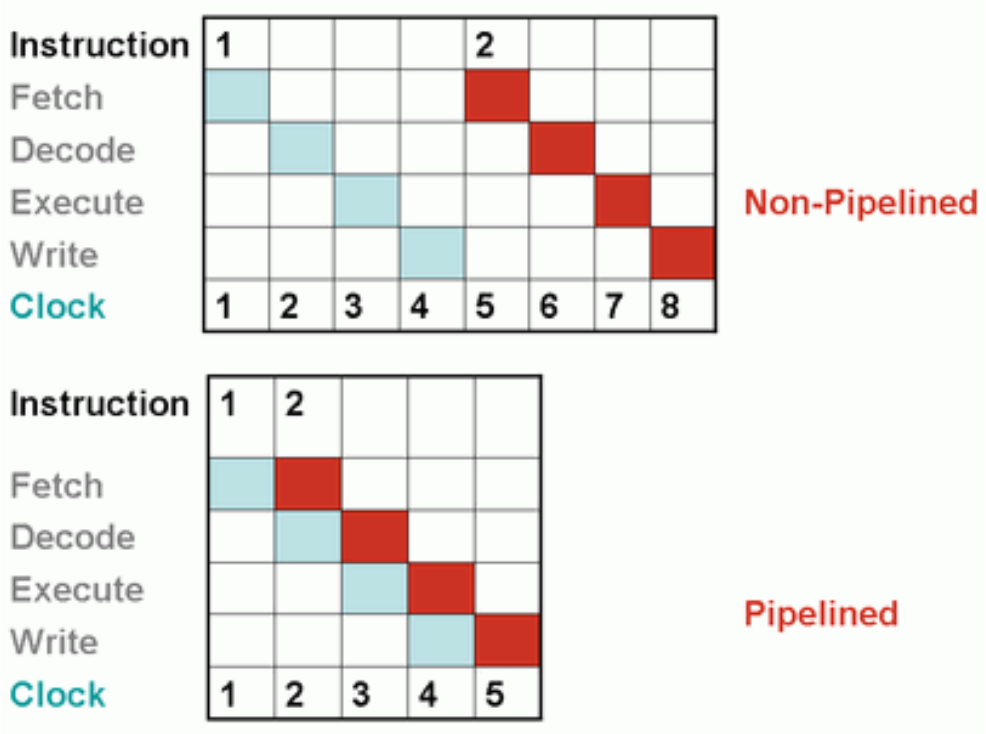
- Each fetch-execute instruction cycle is divided into states, which are one clock pulse long
- Most instructions require multiple steps, and so require **several clock pulses** to complete (multi-cycle processor design)
- Some individual steps (e.g. a **memory access**) take longer & may require additional clock pulses to complete – these clock cycles spent waiting are called **wait states**

System Clock

- The clock speed of a CPU determines how often a new instruction is executed, and is measured in MHz or GHz
  - For example: 1.7GHz means that a computer could executes 1,700,000,000 instructions per second! (if it executes 1 instruction at a cycle)

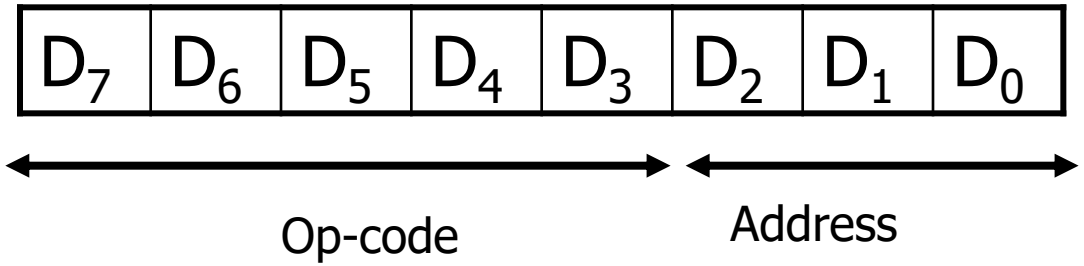
System Clock

- However, all recent microprocessors overlap the fetching, decoding and execution of a number of instructions at the same time – this is called **pipelining**
- Therefore, clock speed is not necessarily an accurate measure of performance, and other measurements are required



Single Word, 1-Address Instruction Format

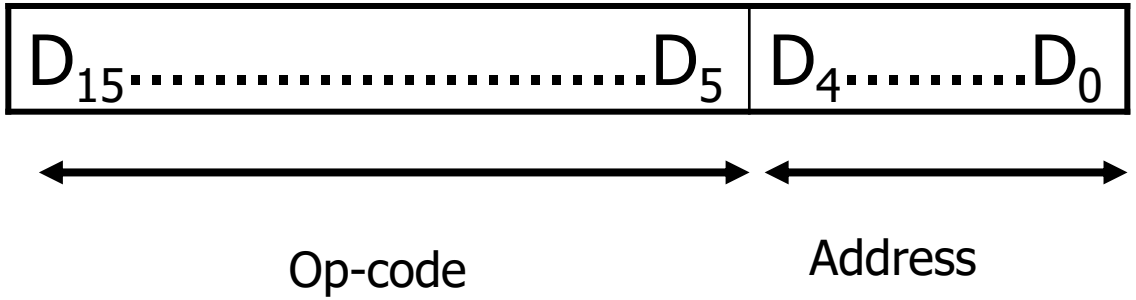
- Single word instruction example for 8-bit words



32 possible op-codes
8 possible addresses

Plausible for register operations

- Example for 16-bit words

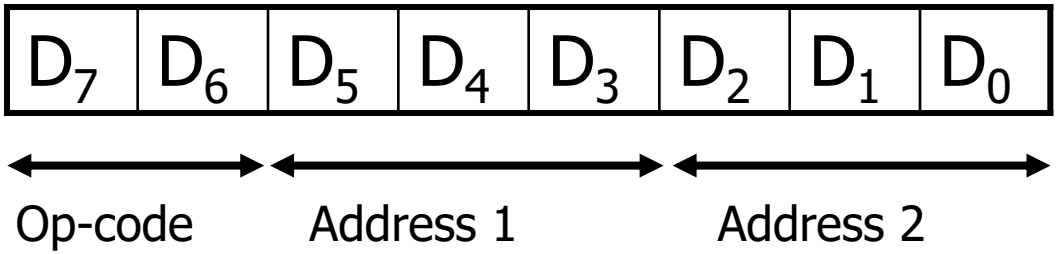


2048 possible op-codes
32 possible adresses

More operation code and addressing possibilities for longer memory words

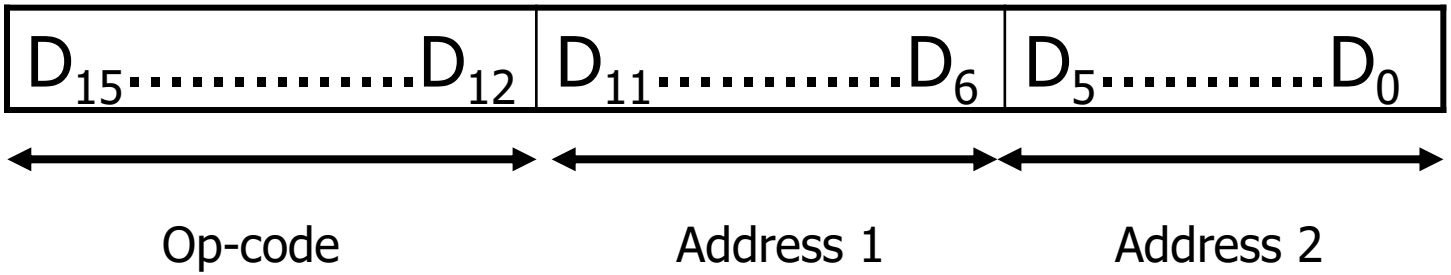
Single Word, 2-Address Instruction Format

- 2-Address instruction in an 8-bit word



4 op-codes
8 Address 1
8 Address 2

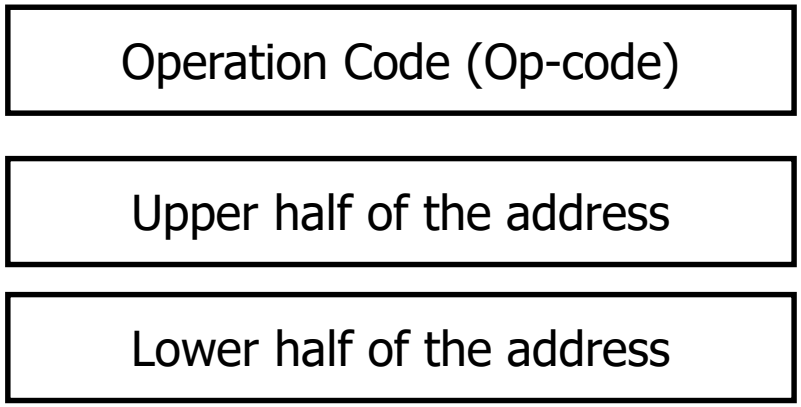
- Example for 16-bit words



16 op-codes
64 Address 1
64 Address 2

Multiple Words, 1-Address Instruction Format

- 1-Address instruction in multiple words



256 Instructions
65536 Address

Instruction Sets

- Depending on the architecture, the instruction set is organized

- CISC (Complex Instruction Set Computer): Contains of a large number of instructions
  - More complex on hardware
  - Examples:
    - MC680x, MC68K, Intel40xx, Intel80xx,
    - Intel x86 (32bit and 64bit laptop, desktop, server systems),
    - IBM System-Z Mainframes and many other supercomputers

- RISC (Reduced Instruction Set Computer): Contains fewer but effective instructions
  - More complex on software
  - Examples:
    - ARM (iPad, iPhone, iPod, Blackberry, Android phones)
    - IBM Power PC (Wii, Xbox, Sony's PS)
    - Oracle (SUN) Sparc
    - Embedded applications
    - Single board computers

Instruction Set Differences

- Consider A = B + C in a high level language

- It might be translated into one instruction with a CISC architecture

```
add mem(B), mem(C), mem(A)
```

- Or four with a RISC architecture

```
load R1, B
load R2, C
add R3, R2, R1
store A, R3
```

Instruction Set Completeness

- A computer should have a set of instructions so that the user can construct machine language programs to evaluate any function that is known to be computable
- Computer design should have a sufficient number of four instruction categories
  - Transfer instructions: Data transfers among registers or registers and main memory
    - Load, Store, Transfer, Swap…
  - Arithmetic, logic, and shift instructions:
    - Add, Complement, Increment, circulate, shift, AND, Clear, Set…
  - Program control Instructions and instructions to check status conditions: Program sequencing and control
    - Compare, Branch, Jump, Go to and Return from Subroutine, Handle Interrupt service, Allow or Not-Allow interrupt requests
  - Input/Output Instructions:
    - Input data, Output data, Control peripherals, Status

Machine and Assembly Language Example:

Assembly language template

{Tag} Operation, Operand : {Explanation}

START LDAA, <$0080> : load ACCA the contents of memory address <$0080>

ADDA, <$0081> : Add ACCA the contents of memory address <$0081>

STAA, <$0082> : Store contents of ACCA to the memory address <$0082>

Address	Content (Machine language)
0010	00 20 00 80
0014	03 20 00 81
0018	01 20 00 82

Instruction Efficiency Example: Greatest common divisor

Euclidean Algorithm for GCD:

Proof of GCD(B,C) evenly divides A:

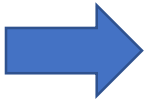
B+C=A

M·GCD(B,C) + N·GCD(B,C) = A

(M + N)·GCD(B,C) = A

C language:

```
int gcd (int i, int j)
{
  while (i != j)
  {
    if (i > j)
       i -= j;
    else
       j -= i;
  }
  return i;
}
```



ARM Assembly language:

```
LOOP:
     CMP    Ri, Rj        ; set condition "NE" if (i != j)
                          ;          "GT" if (i > j),
                          ;       or "LT" if (i < j)
     SUBGT  Ri, Ri, Rj    ; if "GT", i = i-j;
     SUBLT  Rj, Rj, Ri    ; if "LT", j = j-i;
     BNE    LOOP          ; if "NE", then loop
```