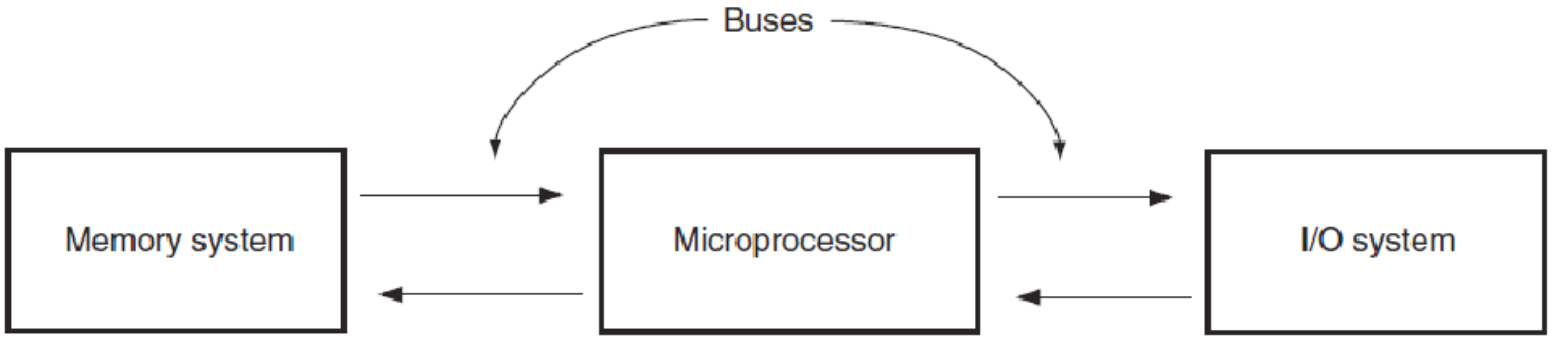
Microprocessors & Interfacing

ARCHITECTURE

BITS Pilani

Dr. Gargi Prabhu
Department of CS & IS

The Microprocessor-based Computer System

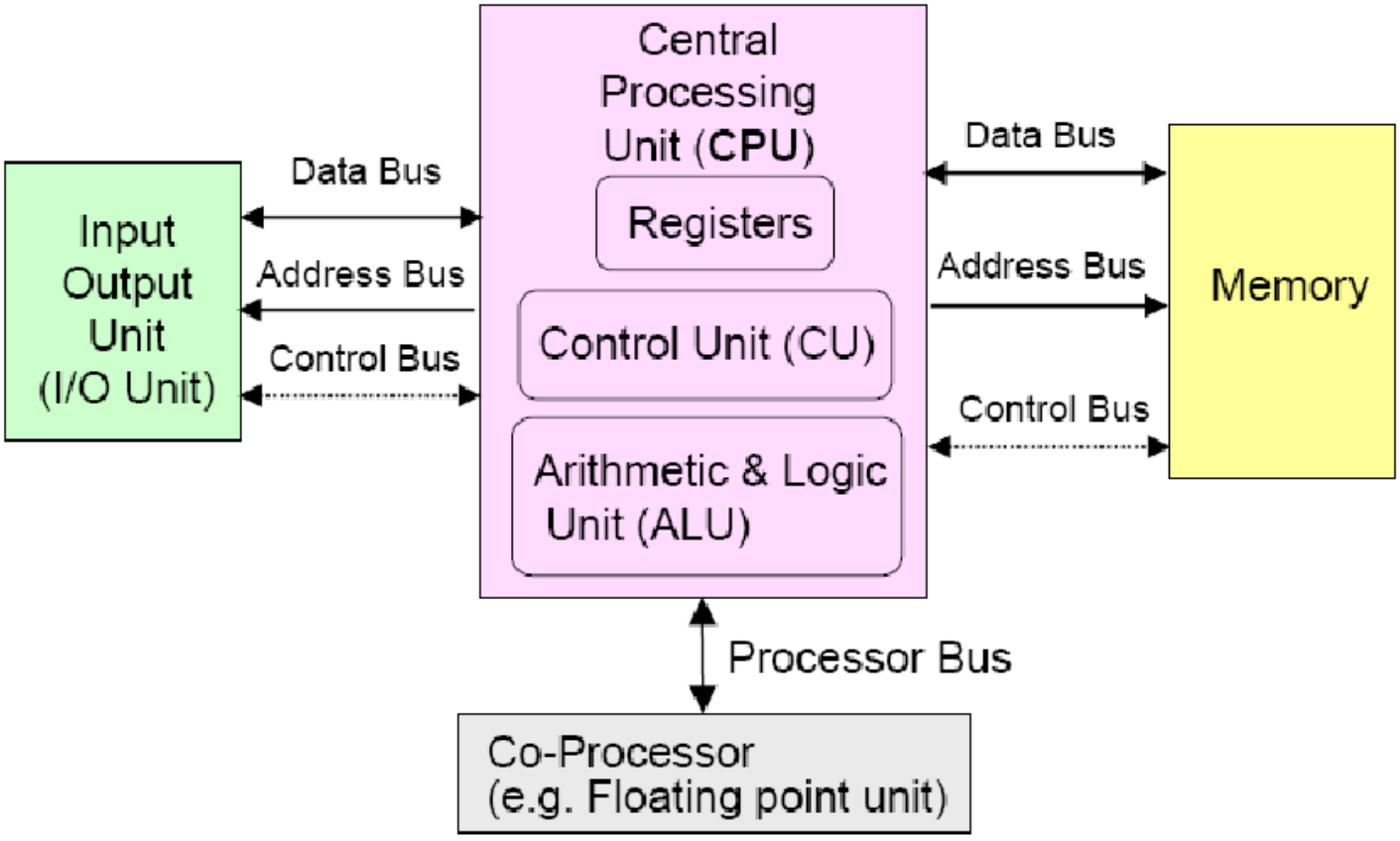


Dynamic RAM (DRAM)
Static RAM (SRAM)
Cache
Read-only (ROM)
Flash memory
EEPROM
SDRAM
RAMBUS
DDR DRAM

8086
8088
80186
80188
80286
80386
80486
Pentium
Pentium Pro
Pentium II
Pentium III
Pentium 4
Core2

Printer
Serial communications
Floppy disk drive
Hard disk drive
Mouse
CD-ROM drive
Plotter
Keyboard
Monitor
Tape backup
Scanner
DVD

The Microprocessor-based Computer System

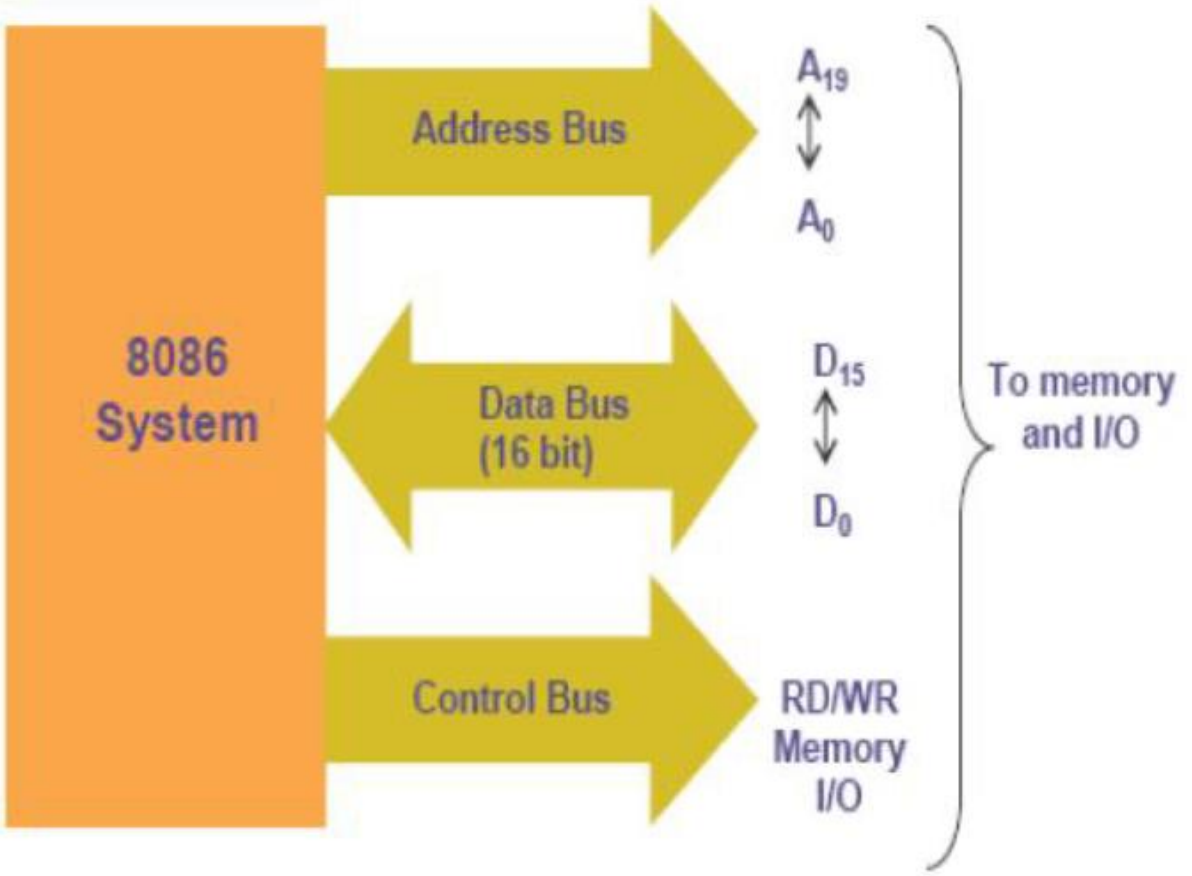


Buses

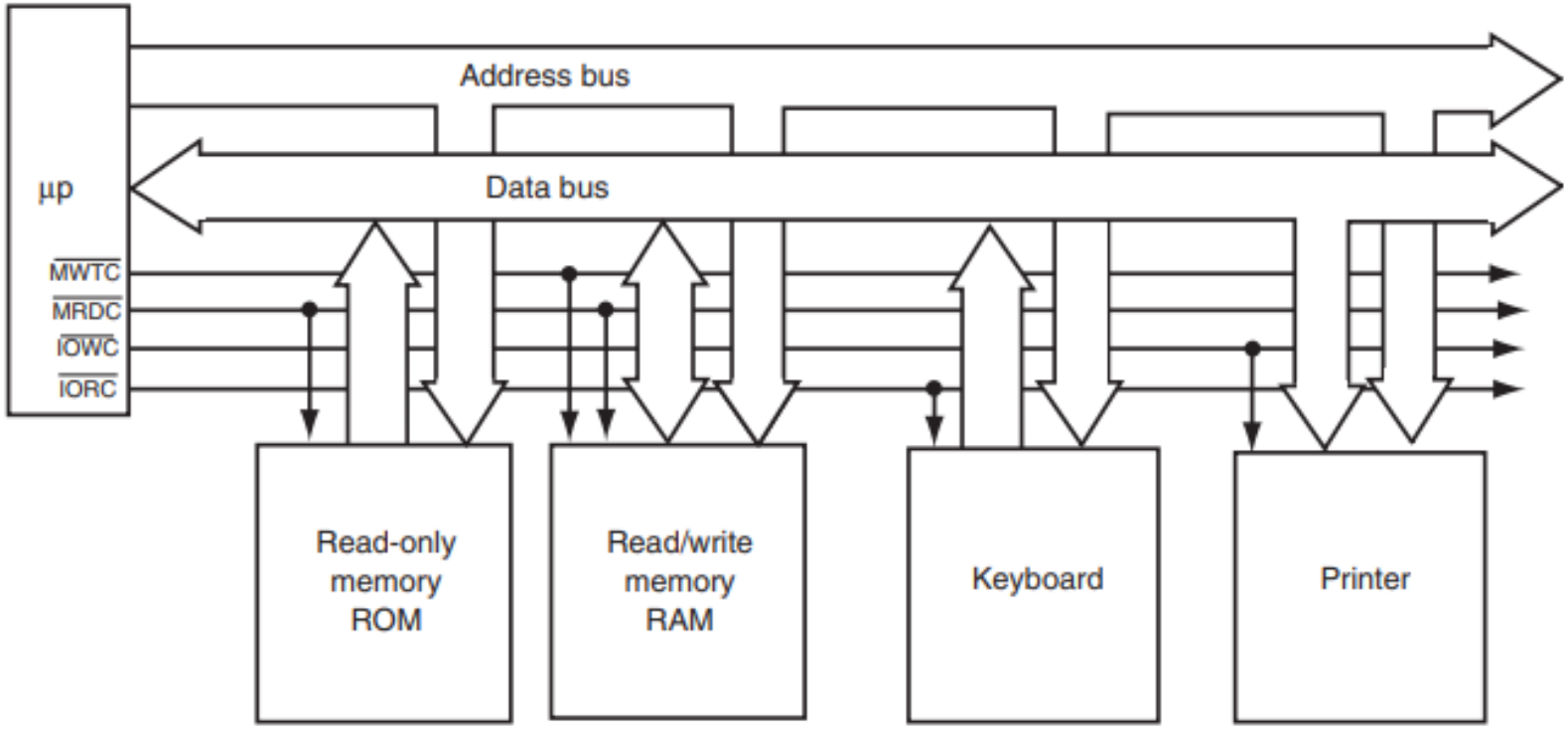
***Address** Bus* provides a memory address to the system memory and I/O address to the system I/O devices

Data Bus transfers data between the microprocessor and the memory and I/O attached to the system

Control Bus provides control signals that cause the memory or I/O to perform a read or write operation



BUSES



Basic control signals: **control signal is active-low**

$\overline{\text{MRDC}}$ (memory read control), **$\overline{\text{MWTC}}$** (memory write control),

$\overline{\text{IORC}}$ (I/O read control), and **$\overline{\text{IOWC}}$** (I/O write control).

CISC Architecture

- **Complex Instruction Set Computer (CISC):** complexity and number of instructions both are increased. Why???
  - Simplify the compilation
- The trend into computer hardware complexity was influenced by various factors, such as
  - To provide support for more customer applications
  - Adding instructions that facilitate the translation from high level language into machine language programs
    - Single machine instruction for each high level language statement
  - Ex: VAX computer, IBM/370 computers, **Intel x86 based processors**, Motorola 68000 Series

CISC Characteristics

- A large number of instructions
- Some instructions for special tasks used infrequently
- A large variety of addressing modes (5 to 20)
- Variable length instruction formats
- Instructions that manipulate operands in memory

- However, it soon became apparent that a complex instruction set has a number of disadvantages
- These include a complex instruction decoding scheme, an increased size of the control unit, and increased logic delays.

In 1980s

The simplicity of the instruction set and addressing modes allows most instructions to execute in a single machine cycle, and the simplicity of each instruction guarantees a short cycle time. In addition, such a machine should have a much shorter design time.

- DAVID A. PATTERSON and CARLO H. SEQUIN

(Ref: RISC I: A Reduced Instruction Set VLSI Computer)

RISC Characteristics

- RISC (Reduced Instruction Set Computing)
- Relatively few instructions
  - 128 or less
- Relatively few addressing modes
  - Memory access is limited to LOAD and STORE instructions
- All operations done within the registers of the CPU
  - Use of overlapped register windows for optimization
- Fixed Length (4 Bytes), easily decoded instruction format
  - Instruction execution time consistent

RISC Processors

- MIPS R4000
  - First commercially available RISC processor
  - Supports thirty-two 64-bit registers
  - 128KB of high speed cache
- SPARC (Sun)
  - Based on Berkeley RISC model
- PowerPC (IBM)
- ARM processor family
- Apple iPods

RISC and CISC Comparison

innovate achieve lead

Example CISC Program

- mov ax, 20
- mov bx, 5
- mul ax, bx

Example RISC Program

```
        mov ax,0
        mov bx, 20
        mov cx,5
again:  add ax, bx
        loop again
```

RISC vs CISC Performance Summary

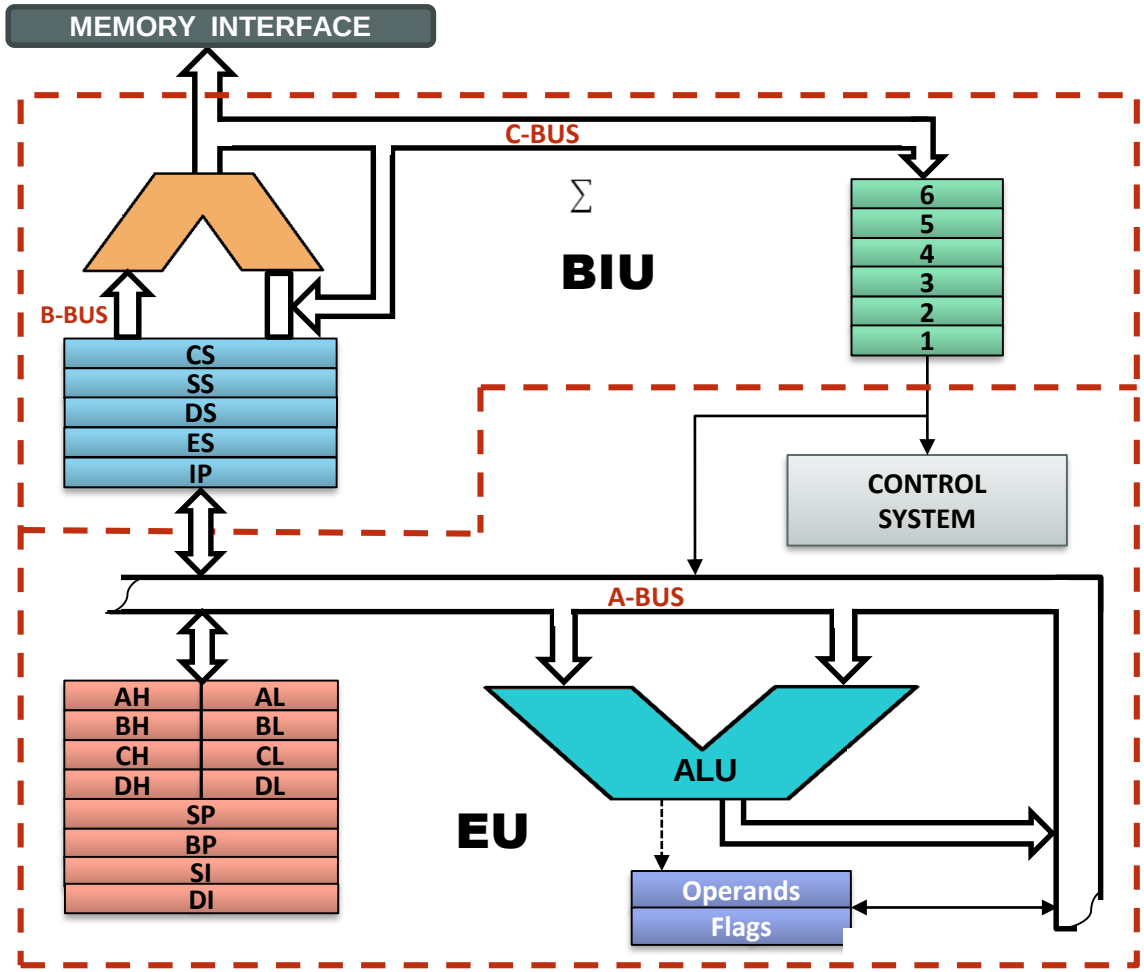
- The CISC approach attempts to minimize the number of instructions per program by sacrificing the number of cycles per instruction.

- RISC does the opposite, reducing the cycles per instruction at the cost of the number of instructions per program.

The 8086 Microprocessor

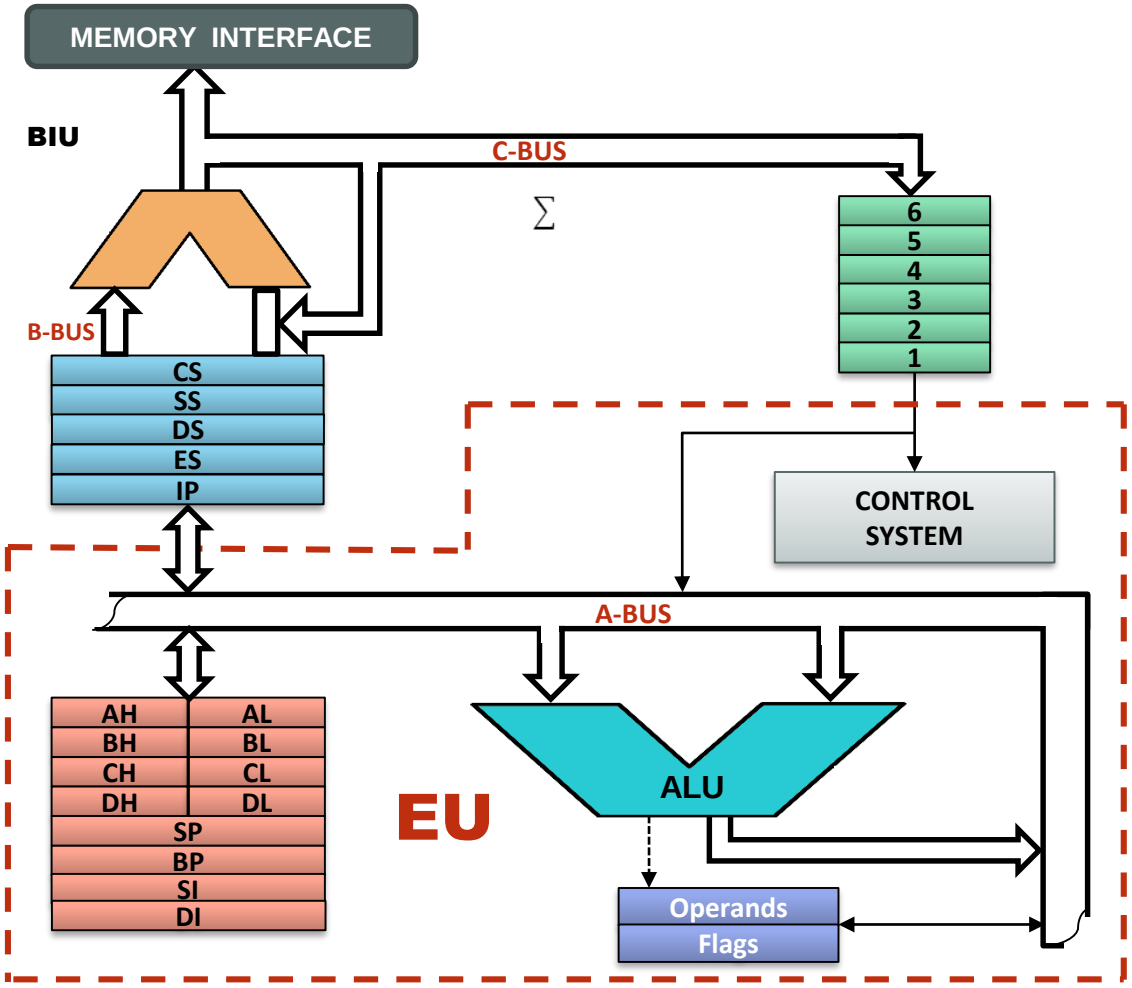
- First 16-bit microprocessor released in 1978
- 20-bit address bus, 1,048,576 memory locations
- 16-bit data bus, read or write 16 bits or 8 bits at a time
- Segmentation of memory for increasing execution speed
- Early implementation of pipelining

Block Diagram of 8086

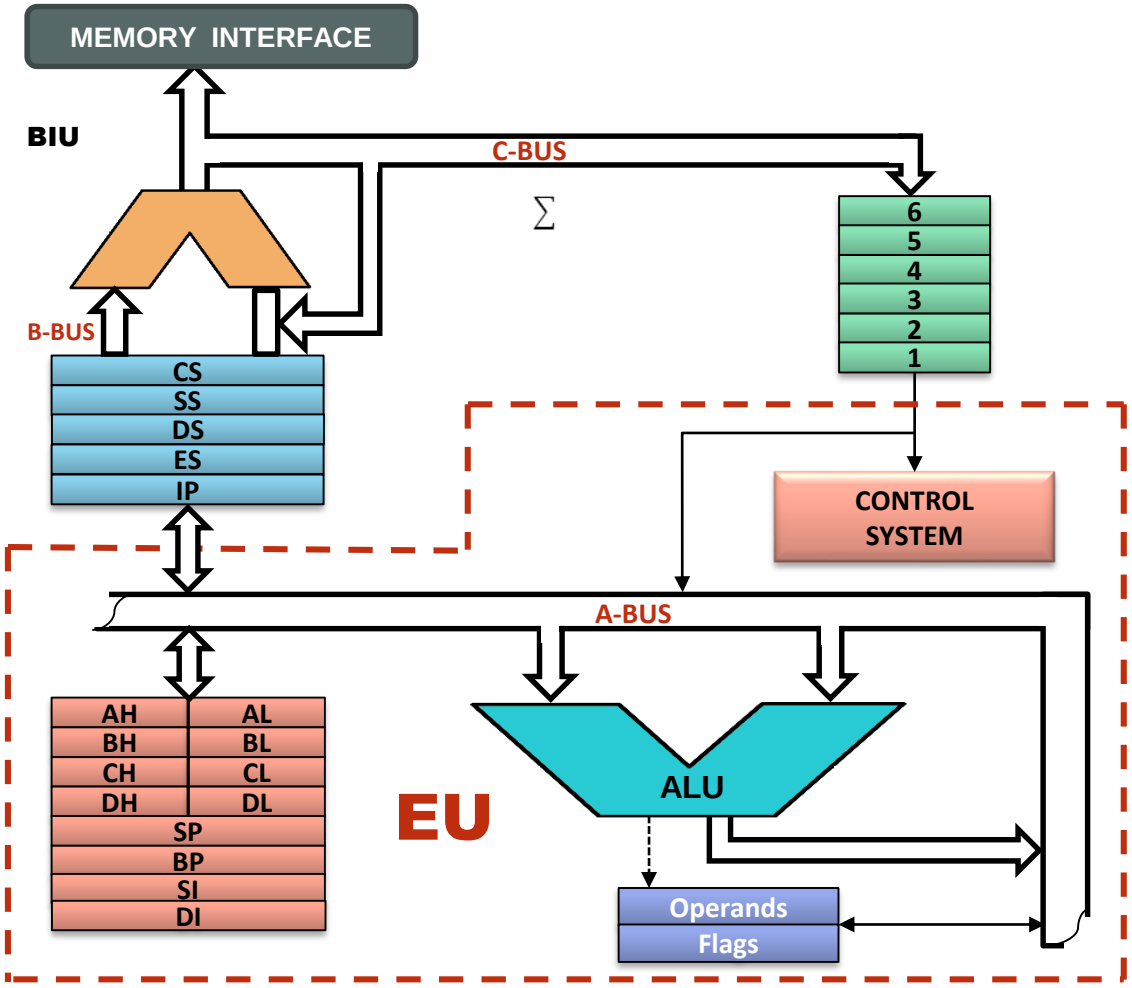


- 8086 CPU is divided into two parts:
  - Bus Interface Unit (BIU)
  - Execution Unit (EU)
- BIU handles all transfers of data and addresses on the buses for EU
- EU tells BIU where to fetch instructions or data from, decodes and executes instruction.

The Execution Unit

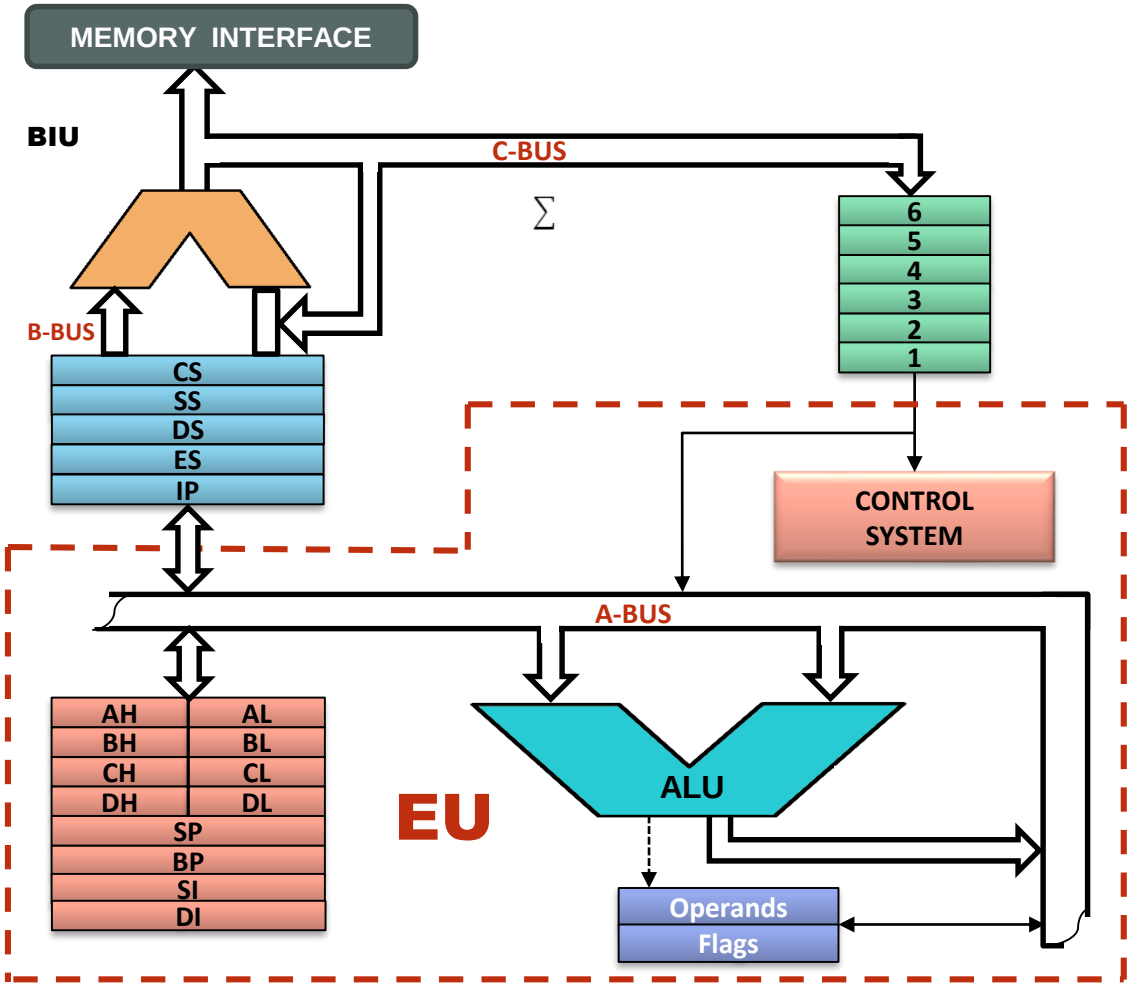


The Execution Unit



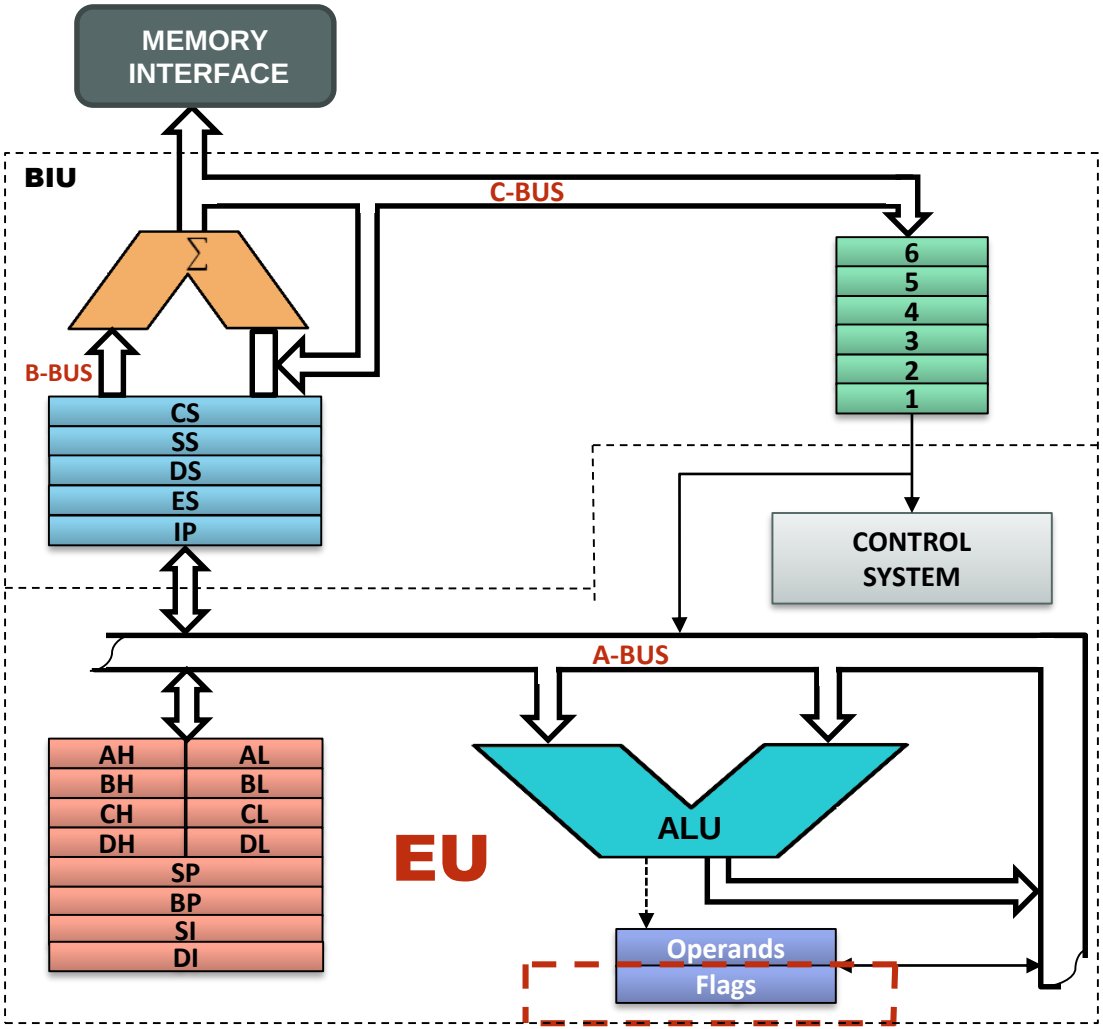
- In control system, Decoder translates instructions fetched from memory
- Control system directs internal operations using control signal

The Execution Unit



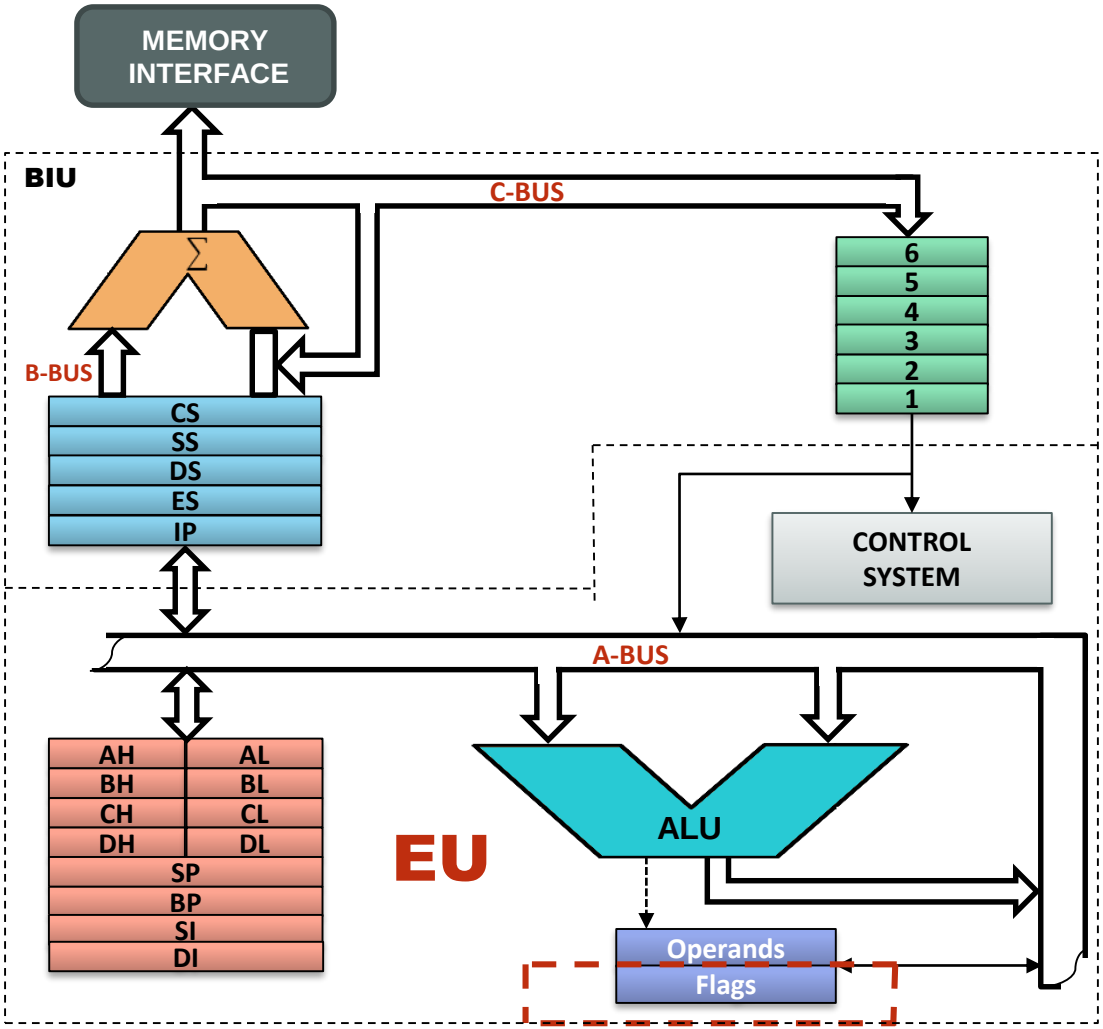
- In control system, Decoder translates instructions fetched from memory
- Control system directs internal operations using control signal
- 16 bit ALU performs operations such as add, sub, AND, OR, XOR

The Execution Unit



- A flag is a flip-flop which indicates some condition produced by the EU
- 8086 features six conditional flags and three control flags

The Execution Unit



Conditional Flag Registers

- Carry flag (CF)
- Parity flag (PF)
- Auxiliary carry flag (AF)
- Zero flag (ZF)
- Sign flag (SF)
- Overflow flag (OF)

Control Flag Registers

- Trap flag (TF)
- Interrupt flag (IF)
- Direction flag (DF)

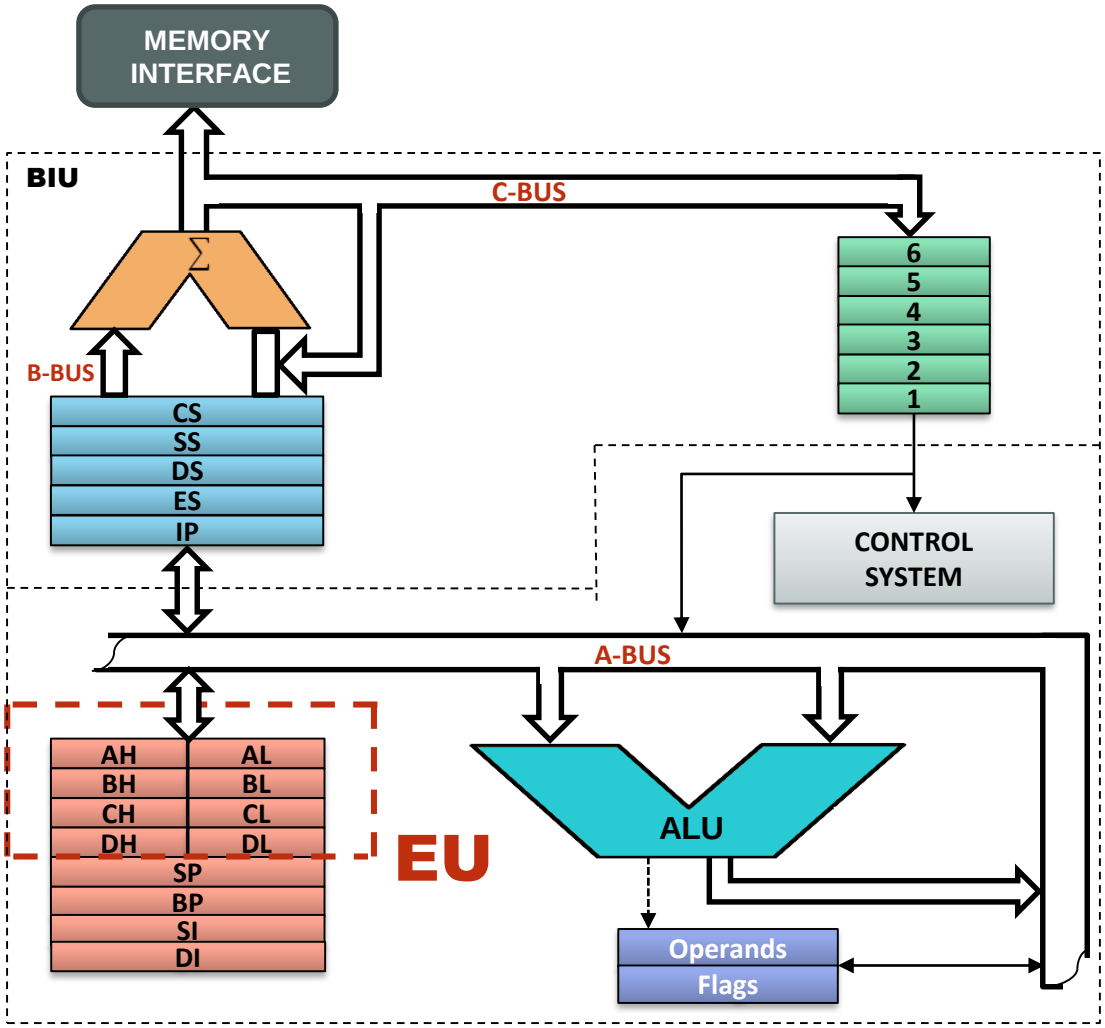
Example of Conditional Flag Registers

```
 0011 0100 1101 1100
+0000 0111 0010 1110
 0011 1100 0000 1010

CF = 0
PF = 1
AF = 1
ZF = 0
SF = 0
OF = 0
```

- **Carry flag (CF)-** carry out of MSB
- **Parity flag (PF) -**set to 1 if low-order 8 bits (low order byte) contain even number of 1's
- **Auxiliary carry flag (AF)** - carry out of bit 3
- **Zero flag (ZF)** - set to 1 if result is 0; set to 0 if result is nonzero
- **Sign flag (SF)** - MSB of result
- **Overflow flag (OF)** - set if carry in to MSB is not equal to carry out from MSB)

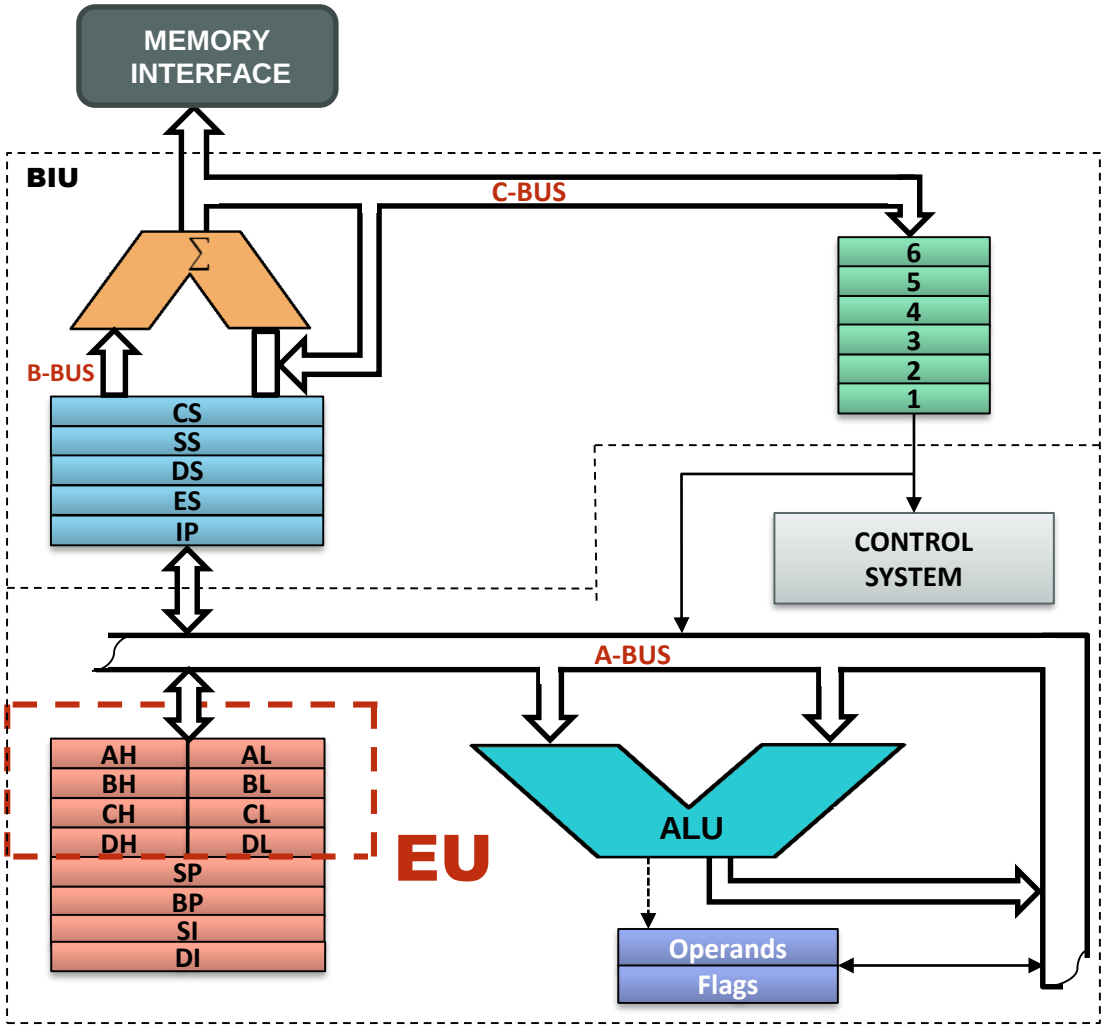
The Execution Unit



General Purpose Registers

- Eight general purpose 8-bit registers
- Can be used individually to store data
- Certain pairs can be used together to store 16-bit data words.
- AL is an accumulator for arithmetic and logic operations.

The Execution Unit



• AX (accumulator register): used for arithmetic and logic operations, and is also used to hold return values from functions.

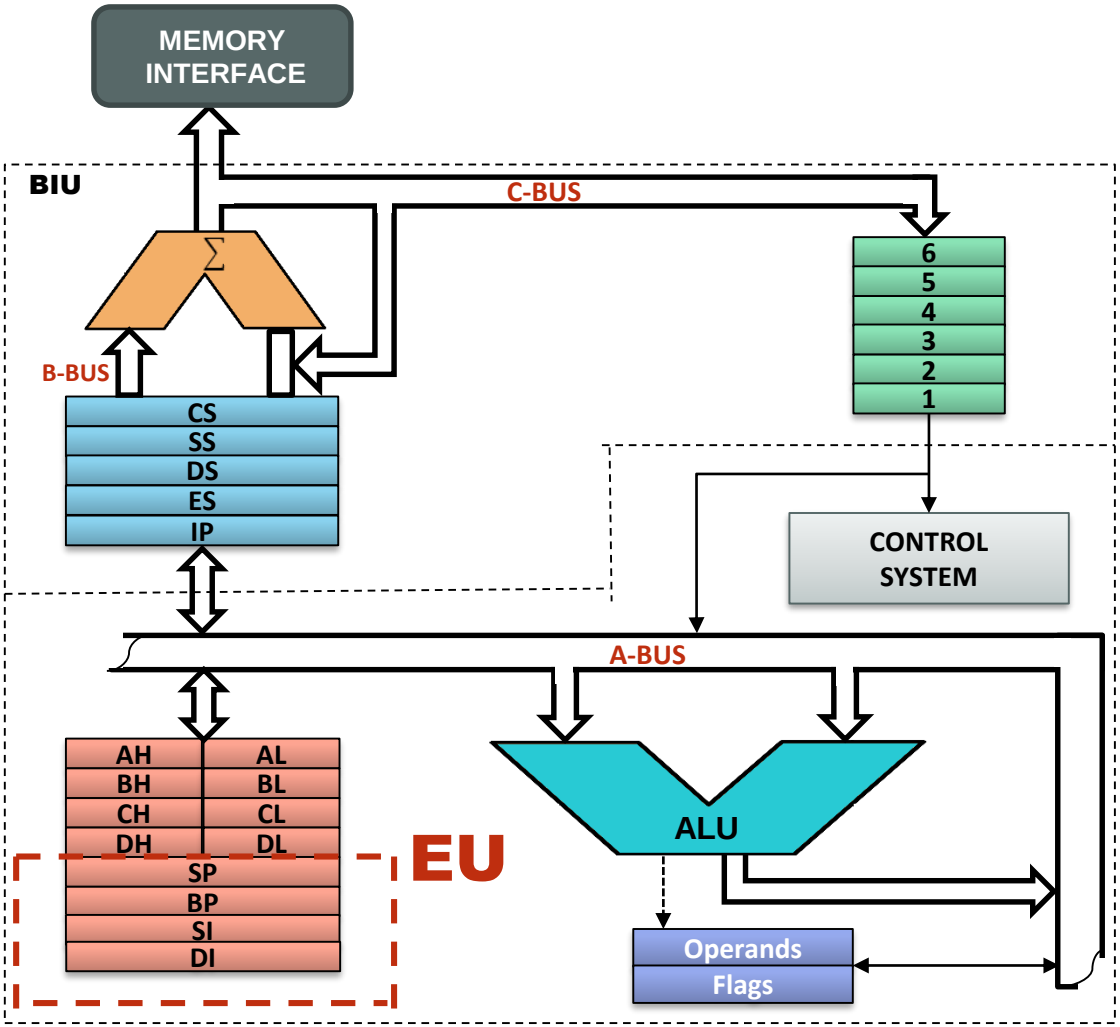
• BX (base register): used as a base pointer for memory access operations, and is also used in indexed addressing modes.

• CX (count register): used as a loop counter for iterative operations, and is also used in shift and rotate instructions.

• DX (data register): used for I/O operations and for holding the high-order bits of some multiplication and division operations.

.

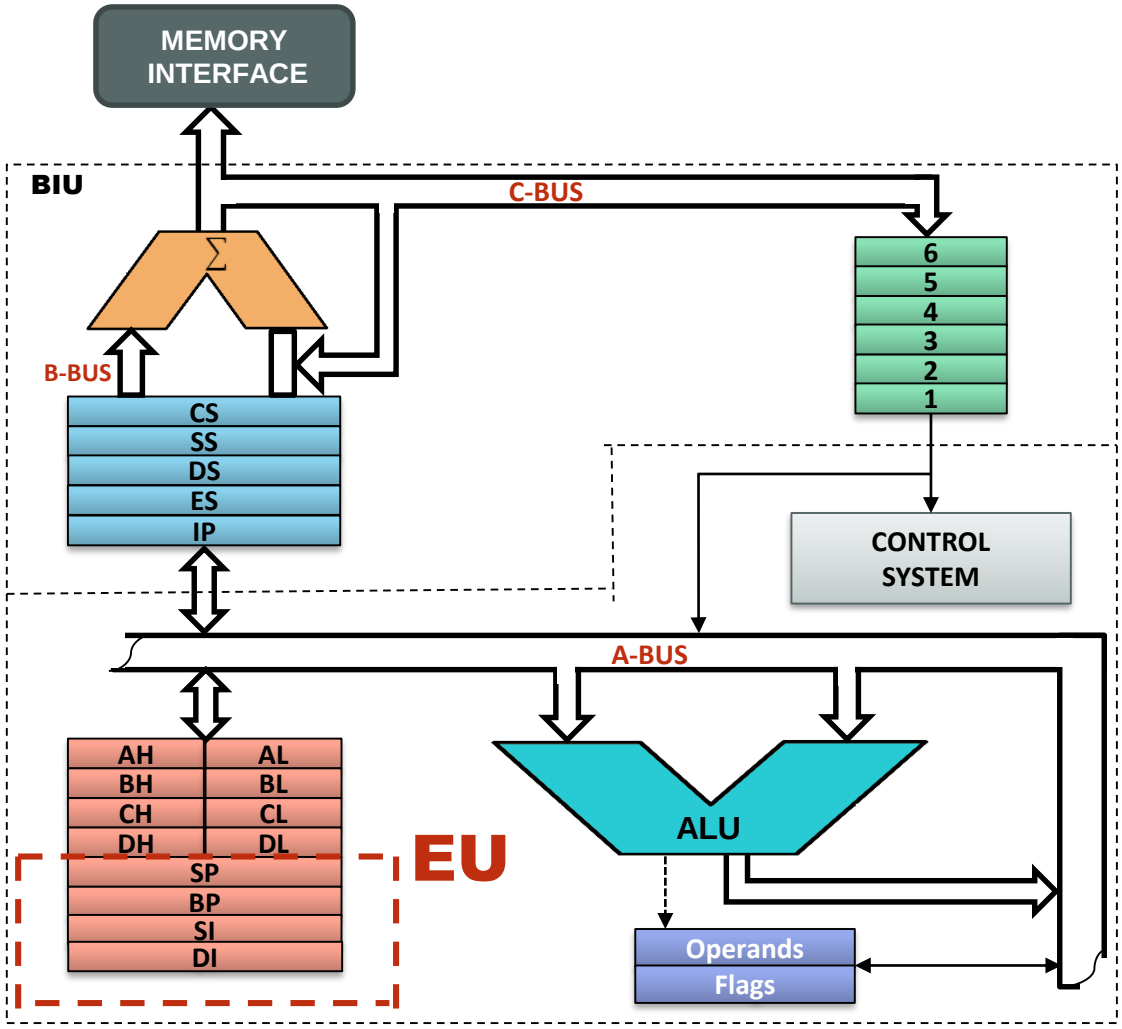
The Execution Unit



Pointer & Index Registers

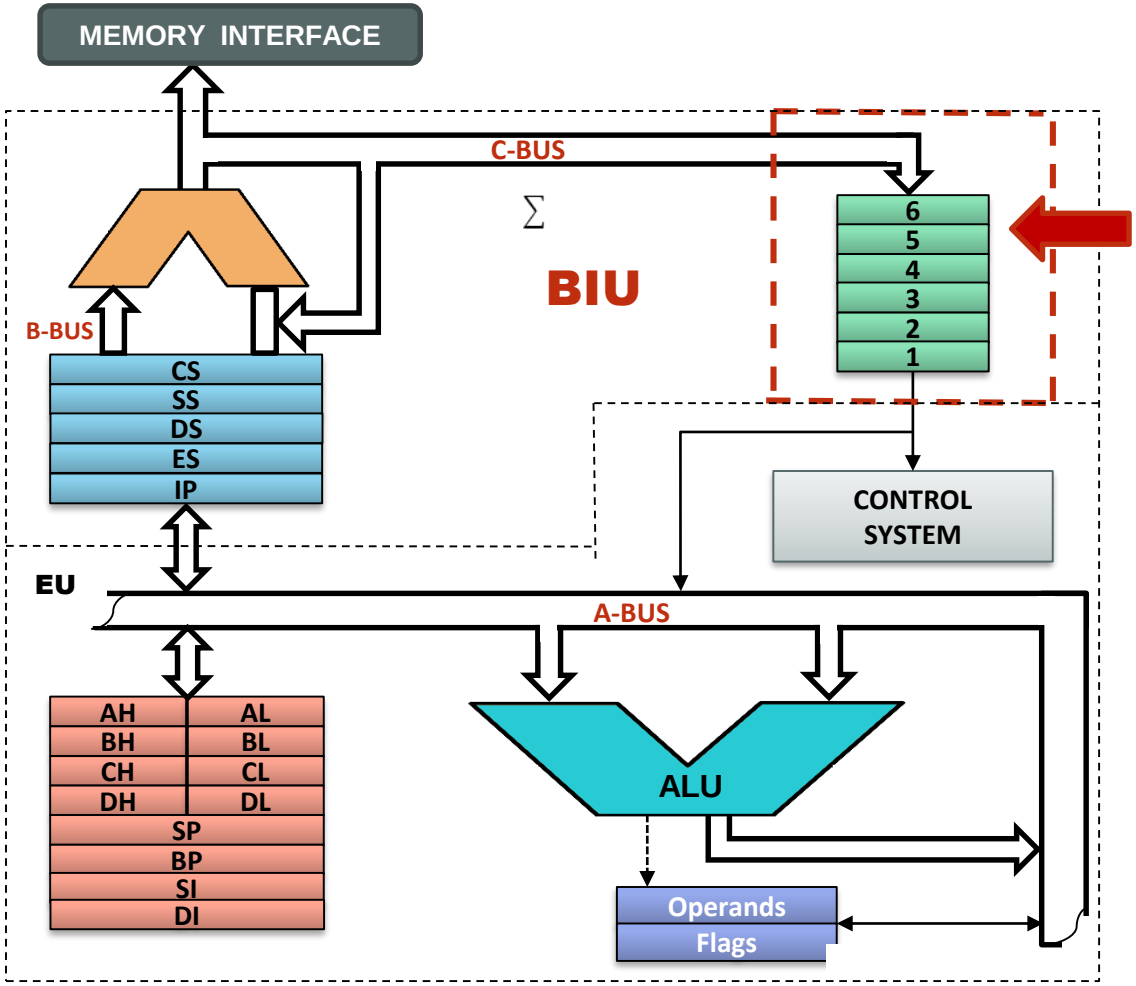
- 64KB Segment is set aside as a stack which stores addresses and data while subprogram is executing.
- Physical address(PA) of stack = SS+SP
- Three more registers: SI, DI and BP which can be used as temporary storage of data
- Main use is to hold 16-bit offset of a data word
  e.g. PA = DS+SI

The Execution Unit



- SP (stack pointer register): used as a stack pointer for push and pop operations, and for accessing function call frames and local variables.
- BP (base pointer register): used as a base pointer for stack operations and for accessing function parameters and local variables.
- SI (source index register): used as a source pointer for memory operations, and is also used in indexed addressing modes.
- DI (destination index register): used as a destination pointer for memory operations, and is also used in indexed addressing modes.

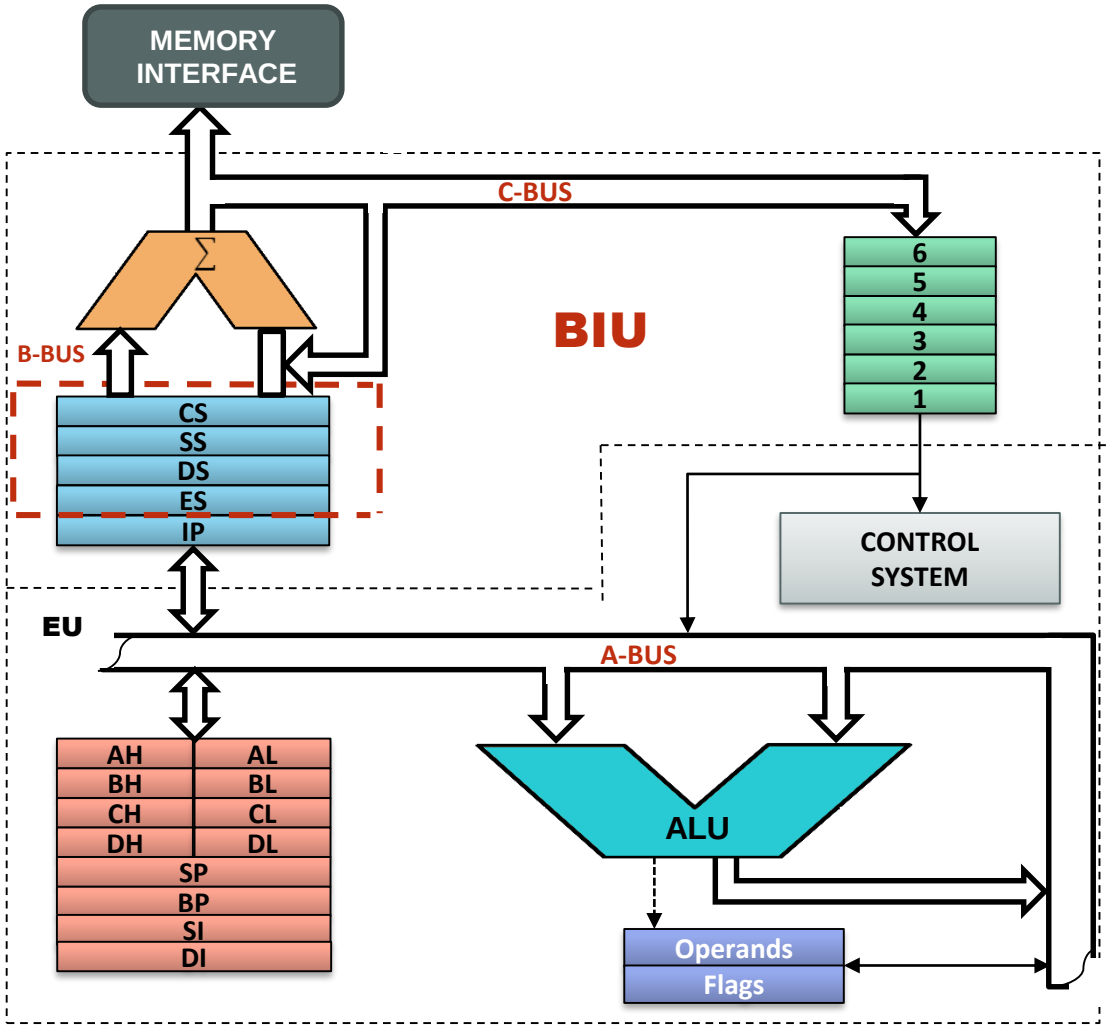
The BIU



Instruction Queue

- BIU fetches up to six instruction bytes and stores in a queue
- Process of fetching next instruction while executing current instruction is called pipelining
- Speeds up the process except for JMP and CALL

The BIU

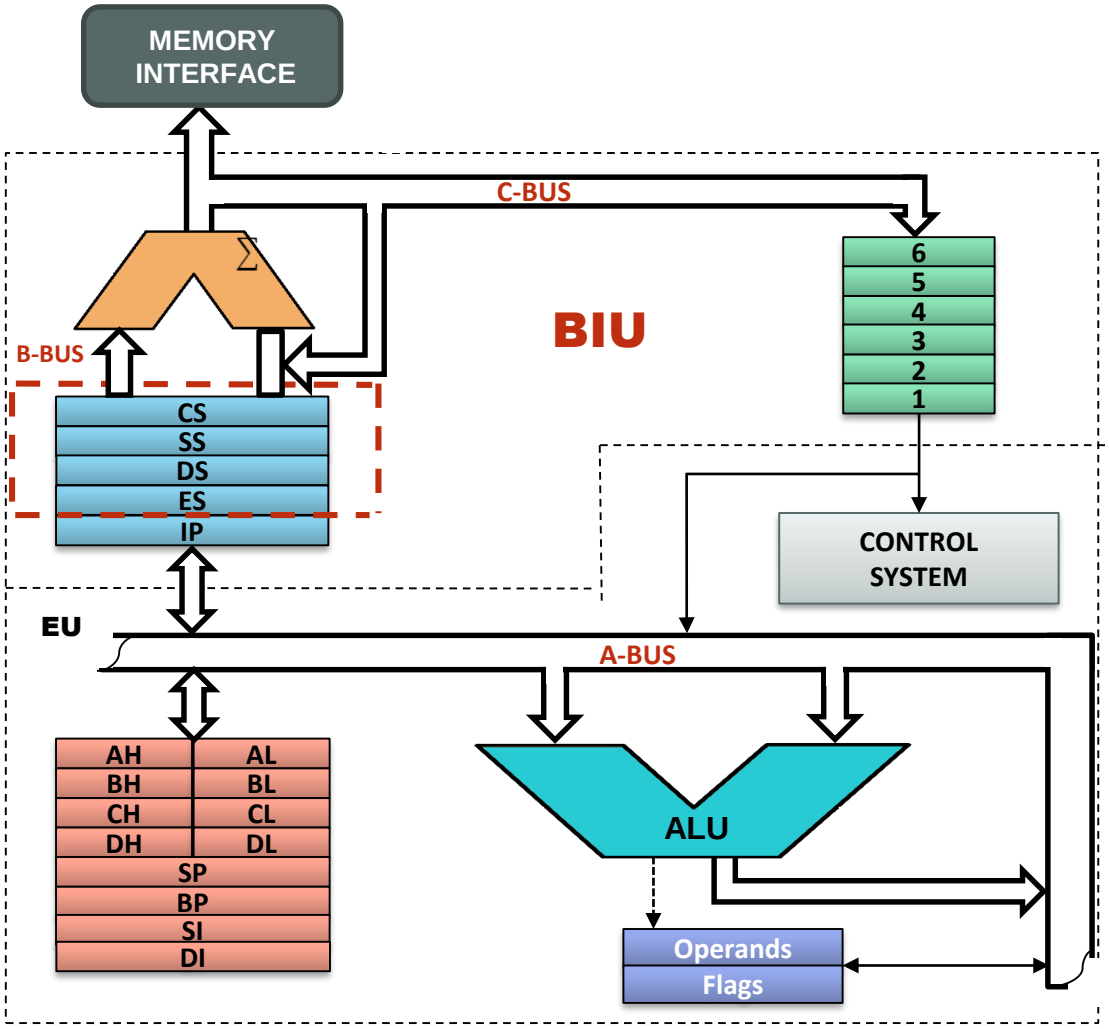


The BIU

The Segment Registers

- BIU sends out 20-bit addresses so it can address any of 2^{20} or 1,048,576 bytes(1MB) in memory.
- Four segment registers hold the upper 16 bits of the starting address of four memory segments
- The use of 16-bit segment registers allows for a maximum of 2^16 (64 KB) segments.
- Each segment can cover a maximum of 64 KB of memory.
- The effective address calculation(segment_register + offset) allows addressing up to 1 MB of memory (2^{20}).

The BIU

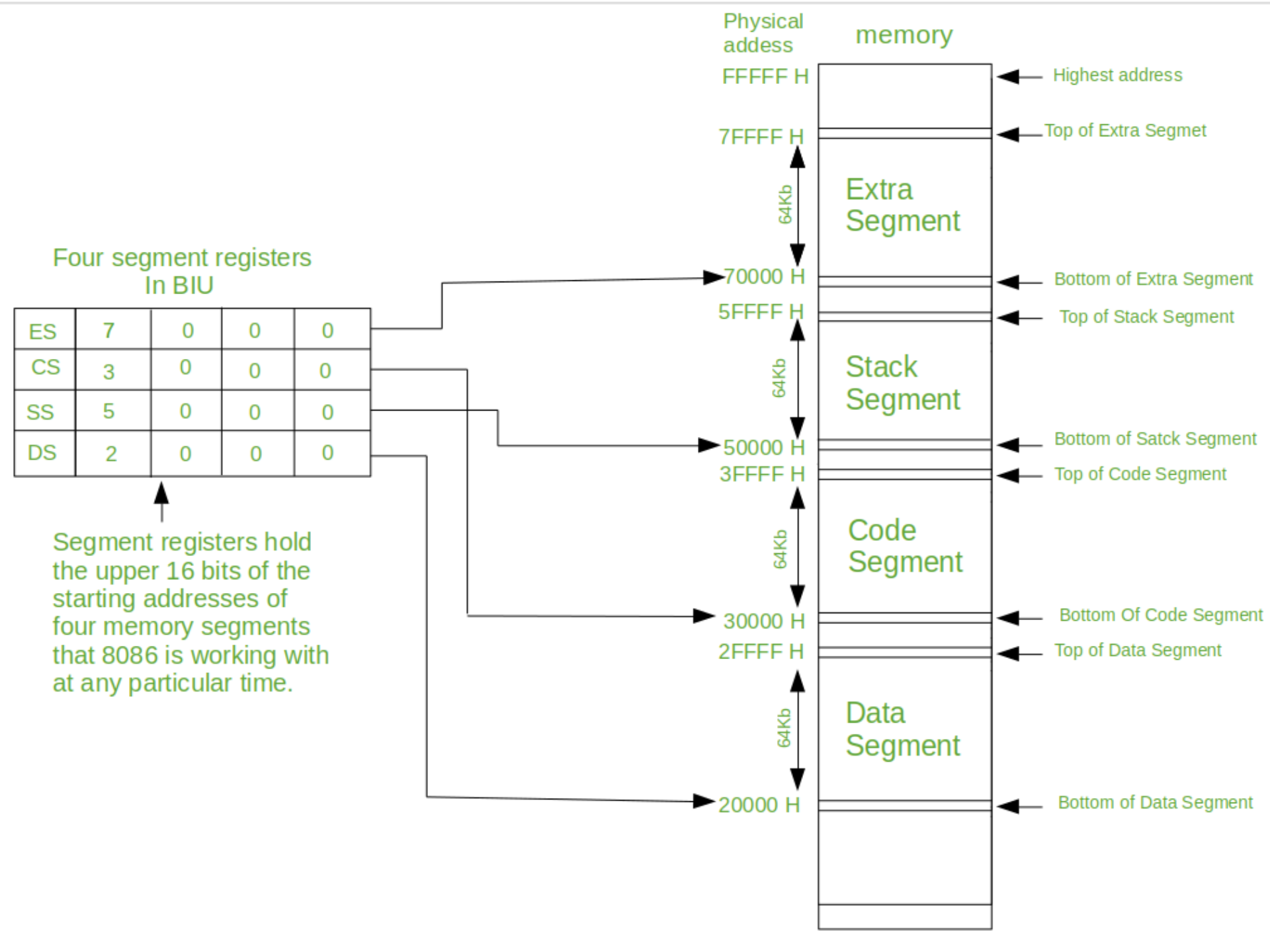


•CS (code segment register): holds the starting address of the code segment where the currently executing program is located.

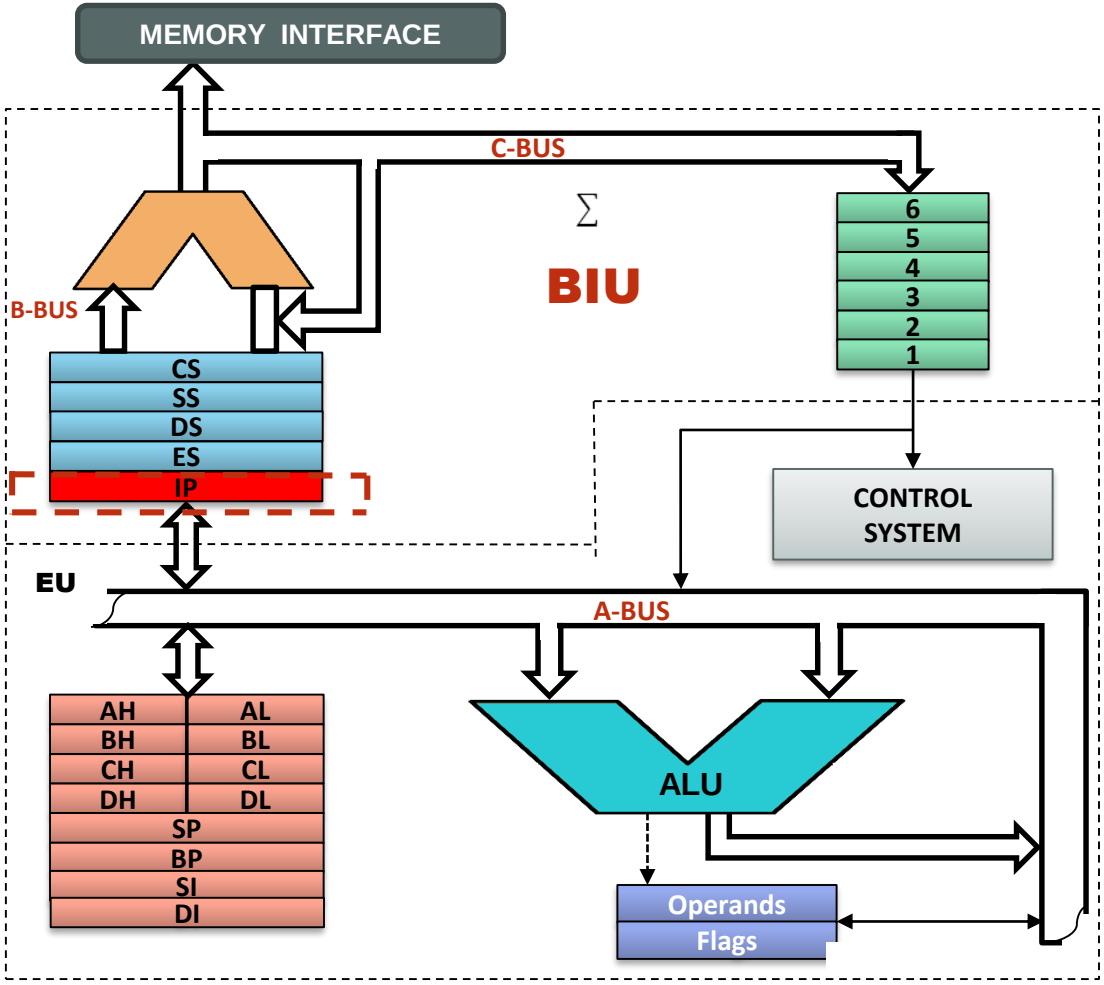
•SS (stack segment register): holds the starting address of the stack segment where the program's stack is located.

•DS (data segment register): holds the starting address of the data segment where the program's variables and arrays are located.

•ES (extra segment register): can be used as an additional data segment register for certain memory operations, such as string operations.

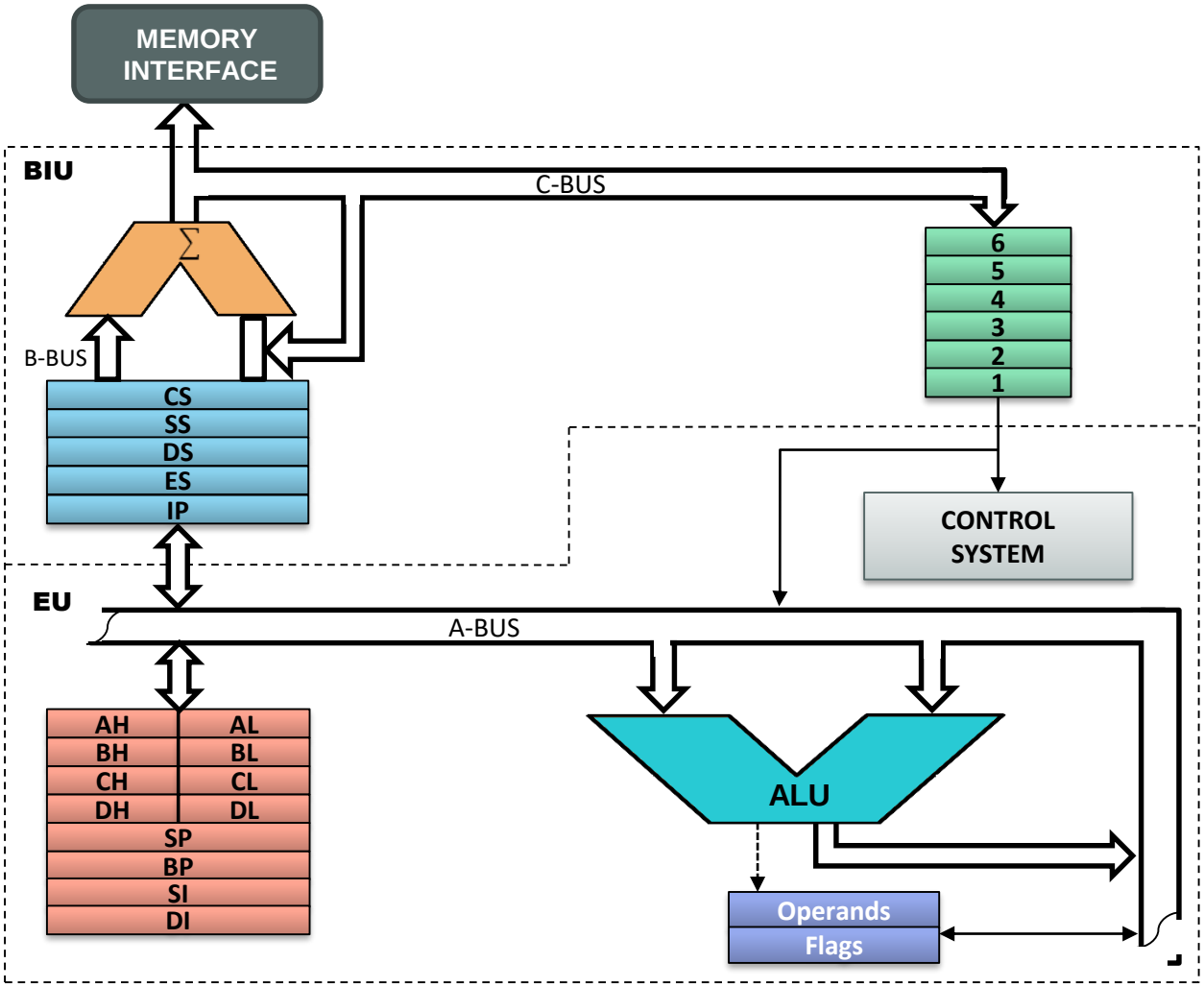

Physical addess
memory
FFFFF H
Highest address
7FFFF H
Top of Extra Segmet
64Kb
Extra Segment
70000 H
Bottom of Extra Segment
5FFFF H
Top of Stack Segment
64Kb
Stack Segment
50000 H
Bottom of Satck Segment
3FFFF H
Top of Code Segment
64Kb
Code Segment
30000 H
Bottom Of Code Segment
2FFFF H
Top of Data Segment
64Kb
Data Segment
20000 H
Bottom of Data Segment
Four segment registers In BIU
ES 7 0 0 0
CS 3 0 0 0
SS 5 0 0 0
DS 2 0 0 0
Segment registers hold the upper 16 bits of the starting addresses of four memory segments that 8086 is working with at any particular time.

The BIU



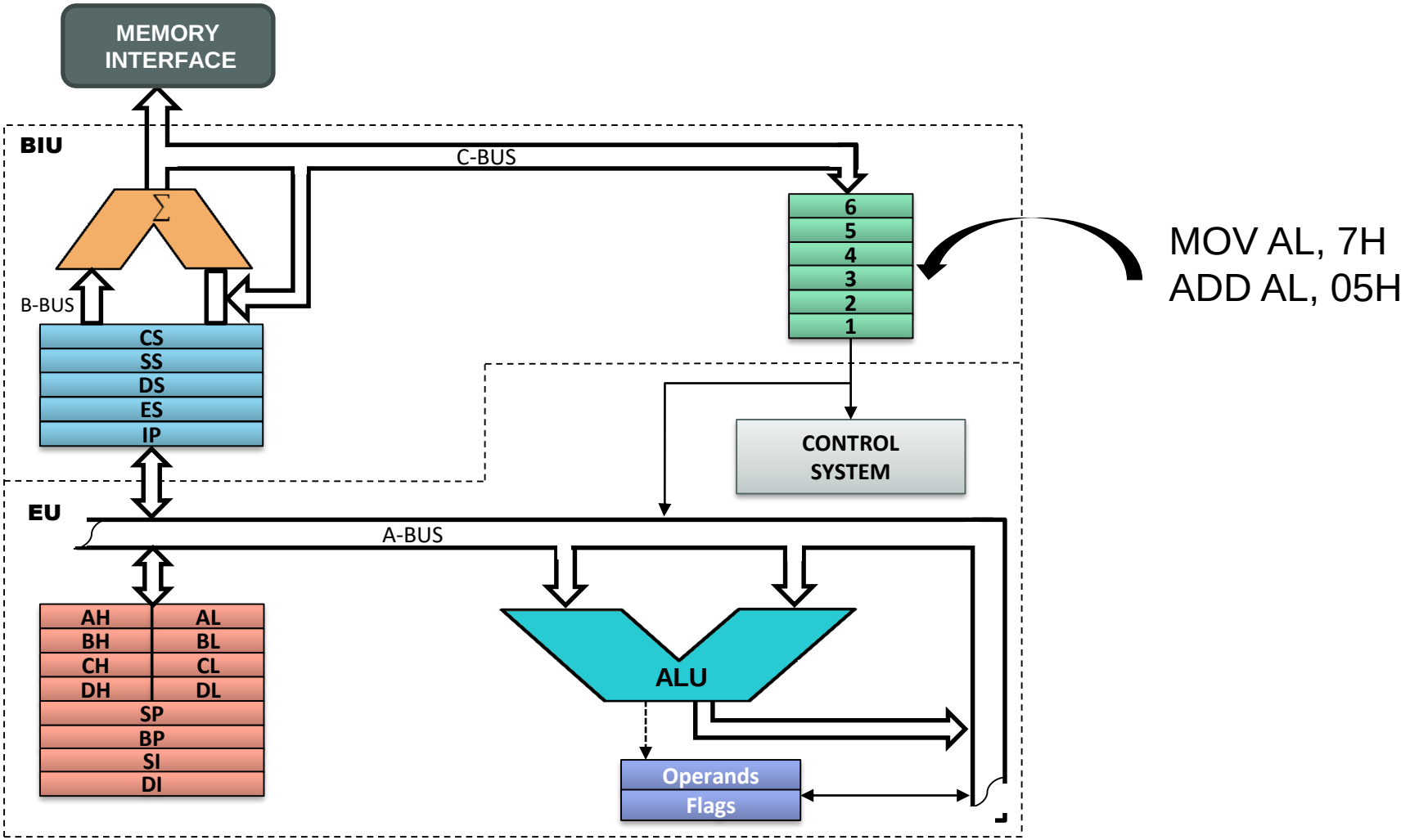
- The instruction pointer (IP) is a 16-bit register in the 8086 microprocessor that points to the memory location of the next instruction to be executed.
- The IP register works in conjunction with the code segment register (CS) to form a pointer to the current instruction in memory.
- The CS register holds the starting address of the code segment, while the IP register holds the offset of the next instruction.

Working of 8086 with an example

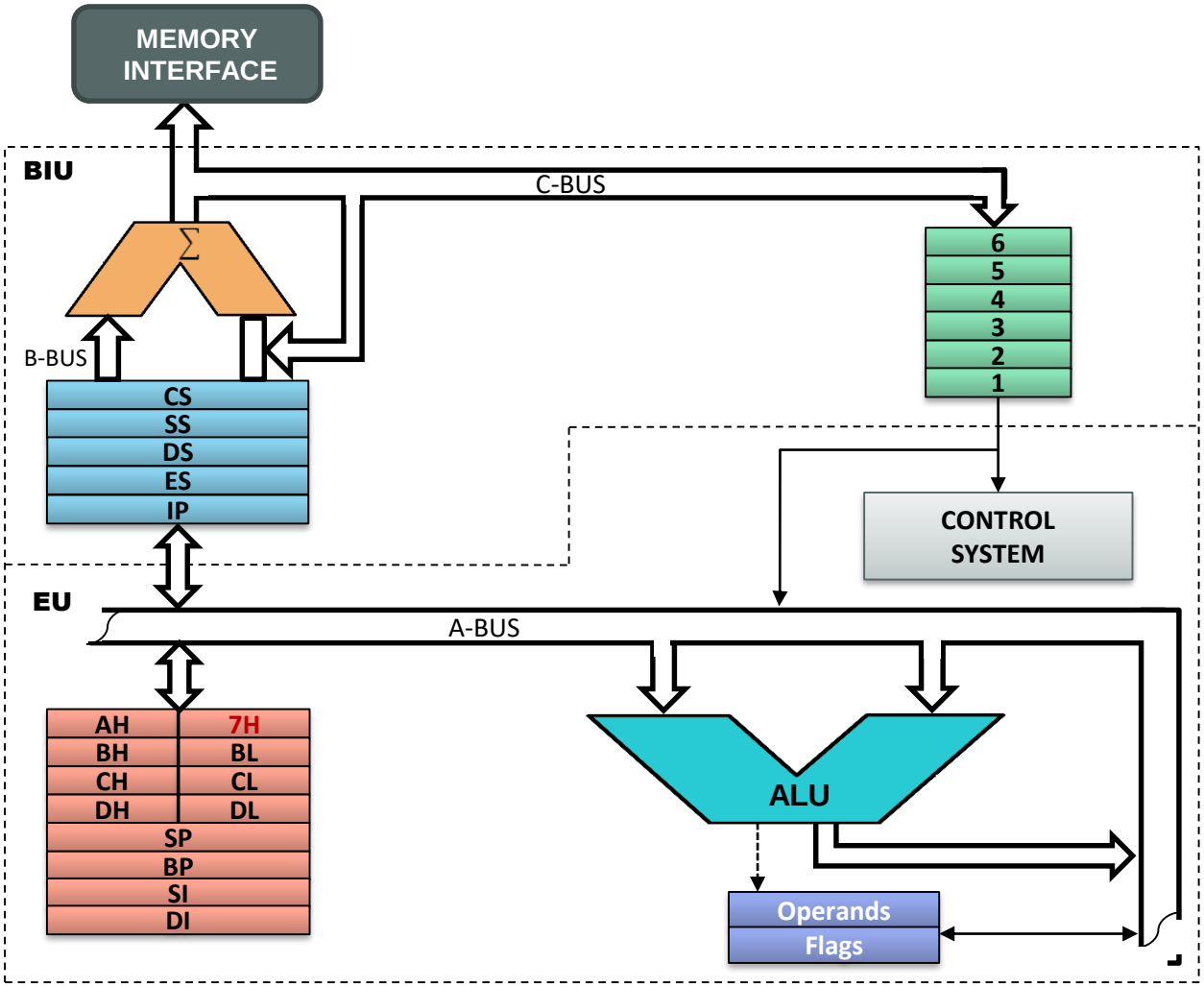


MOV AL, 7H
ADD AL, 05H

Working of 8086 with an example



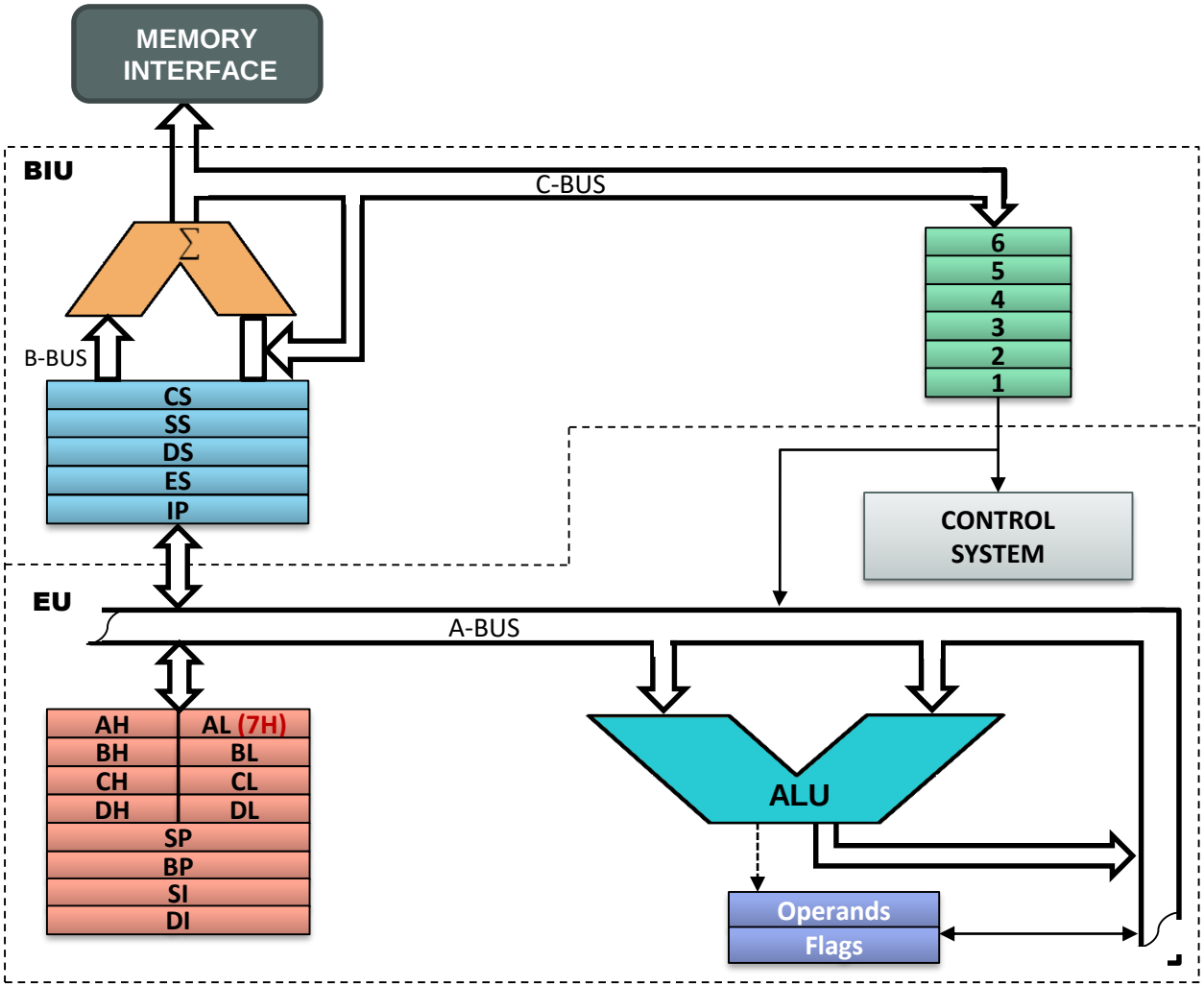
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

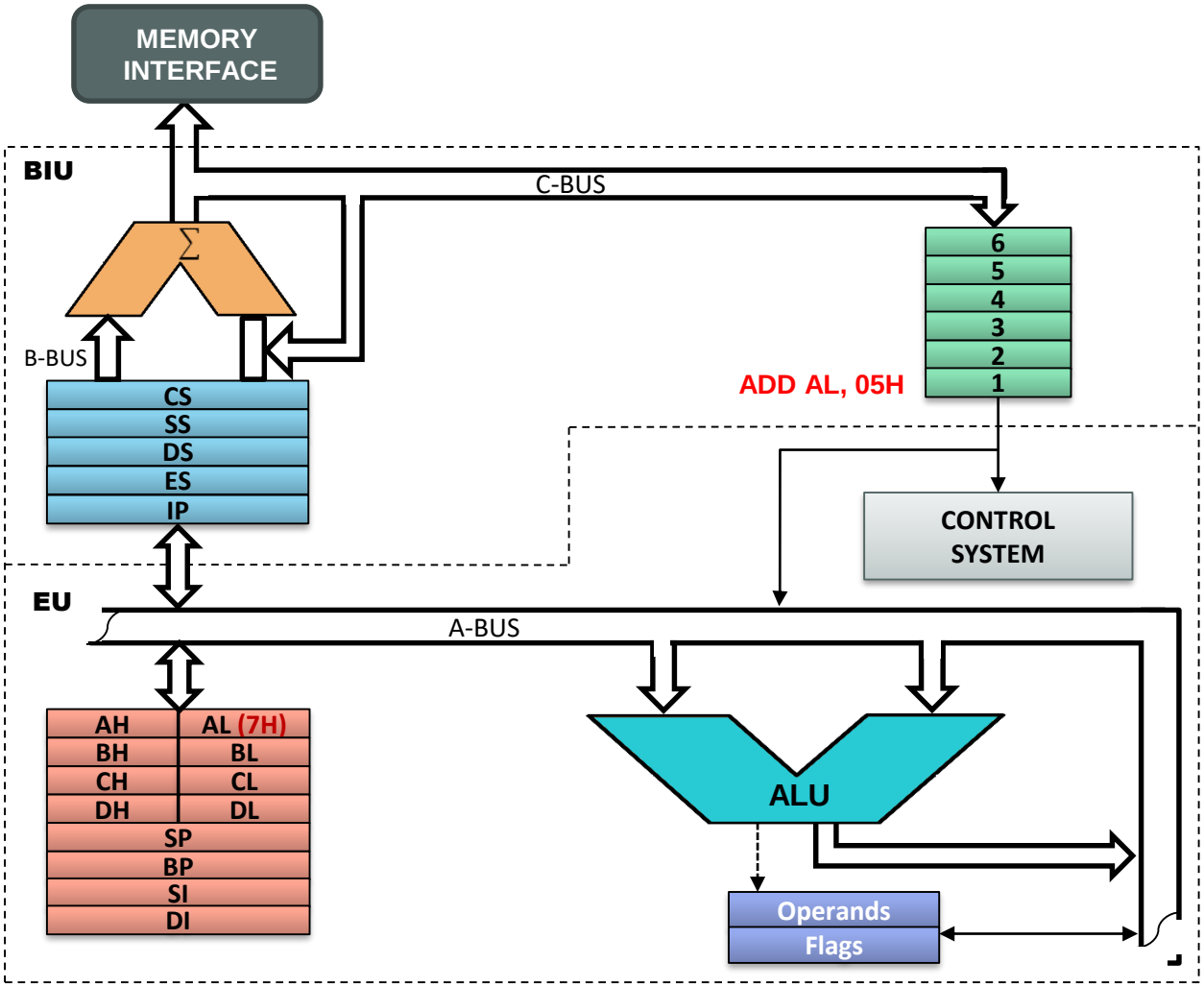
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

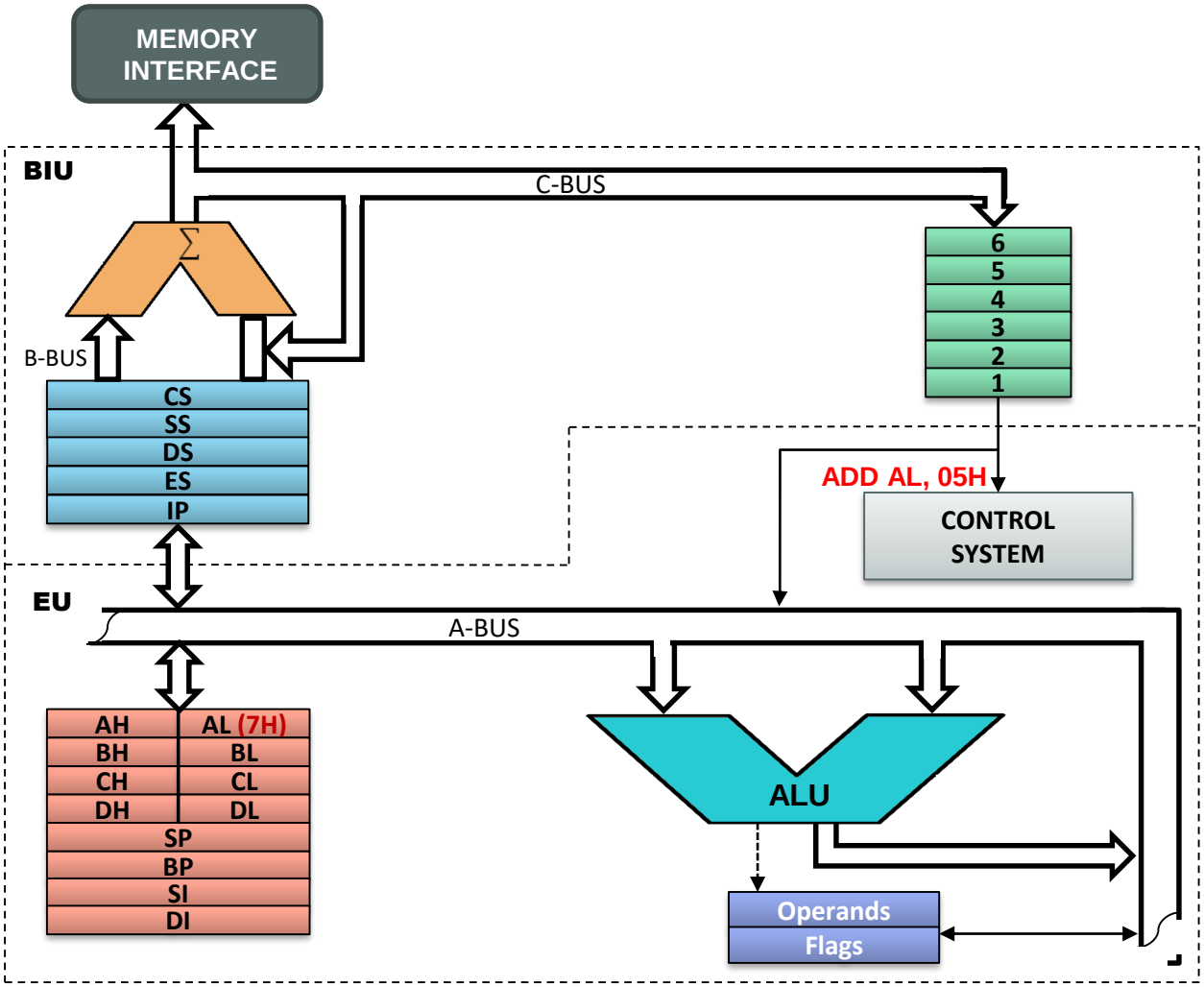
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

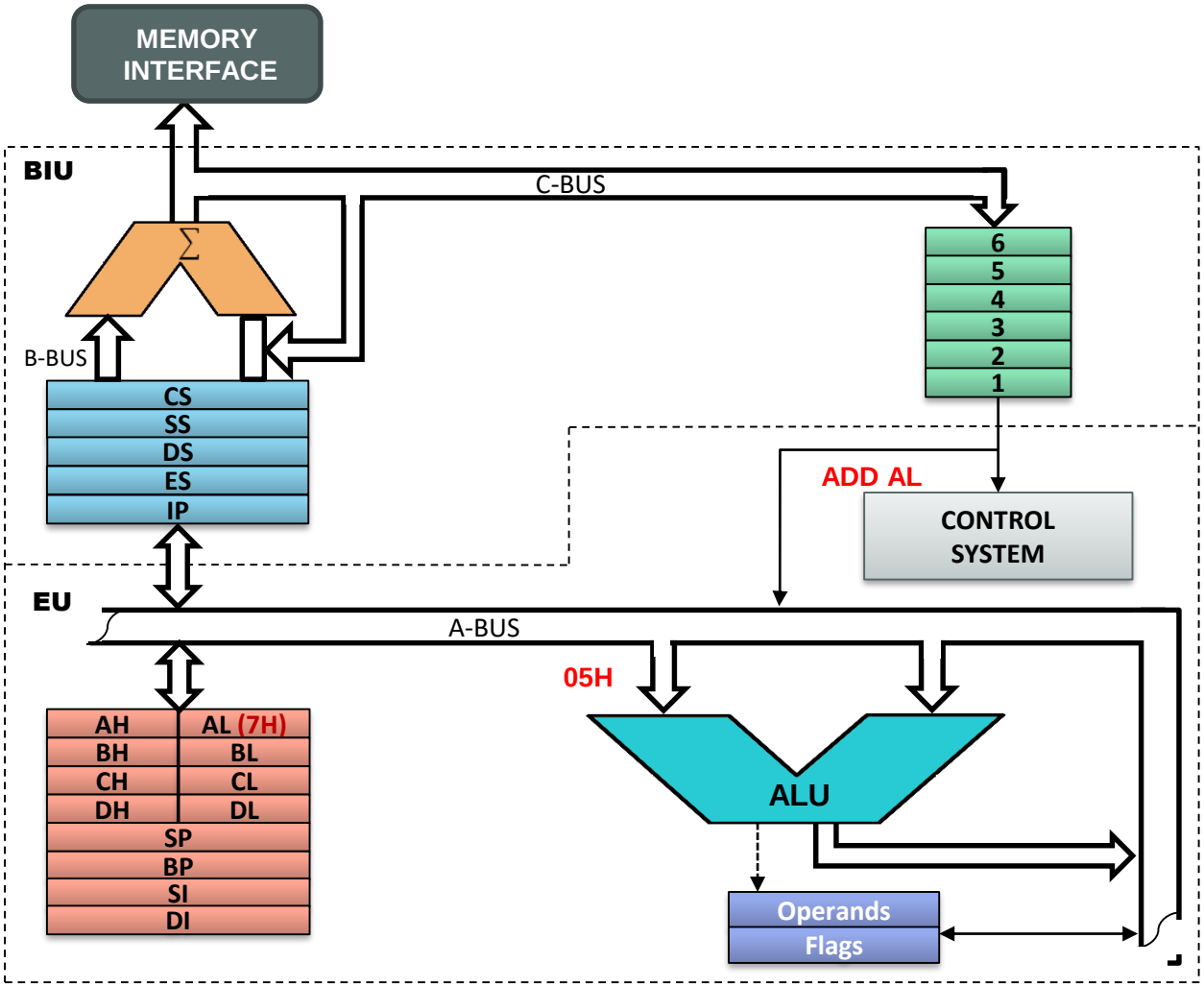
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

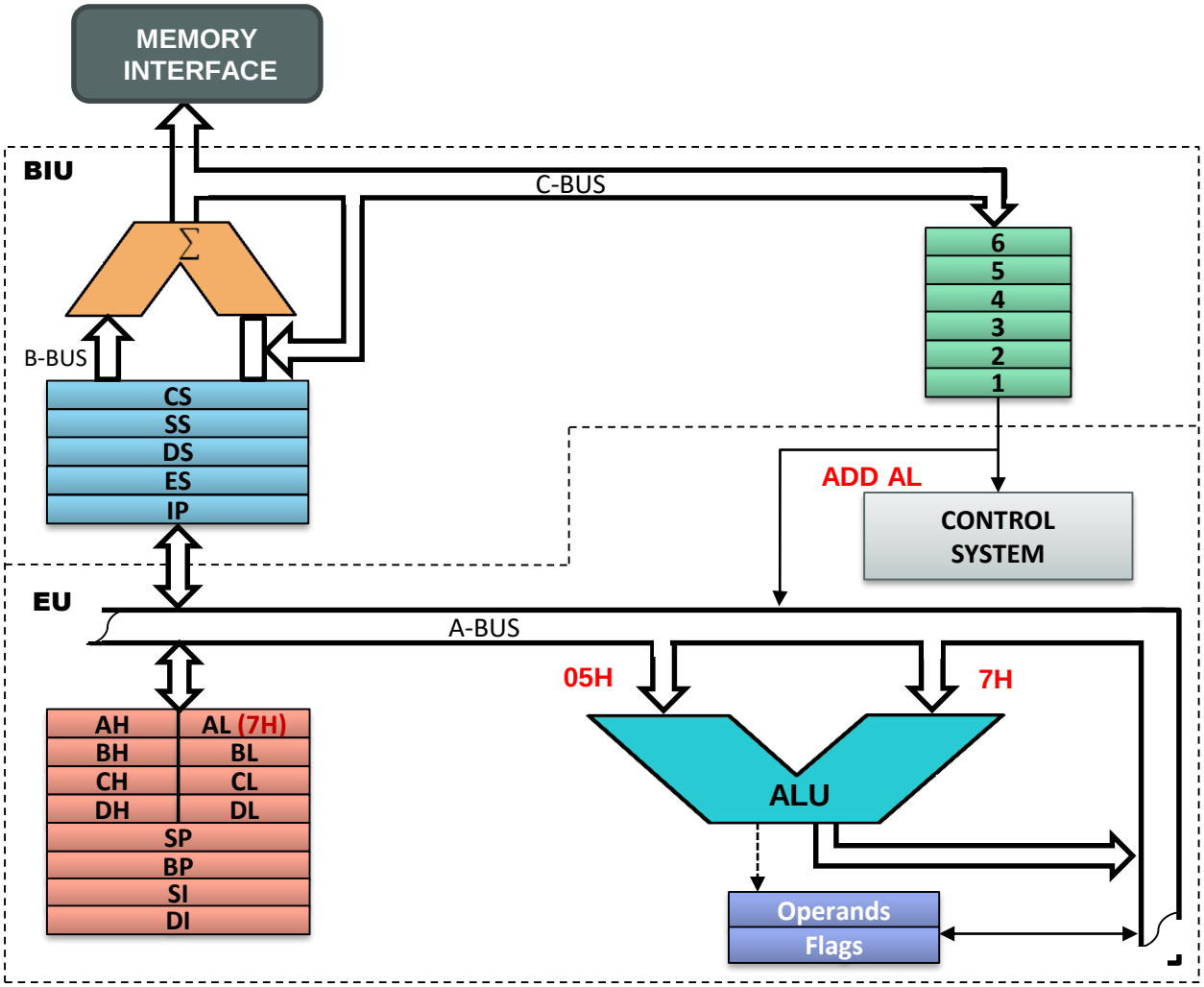
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

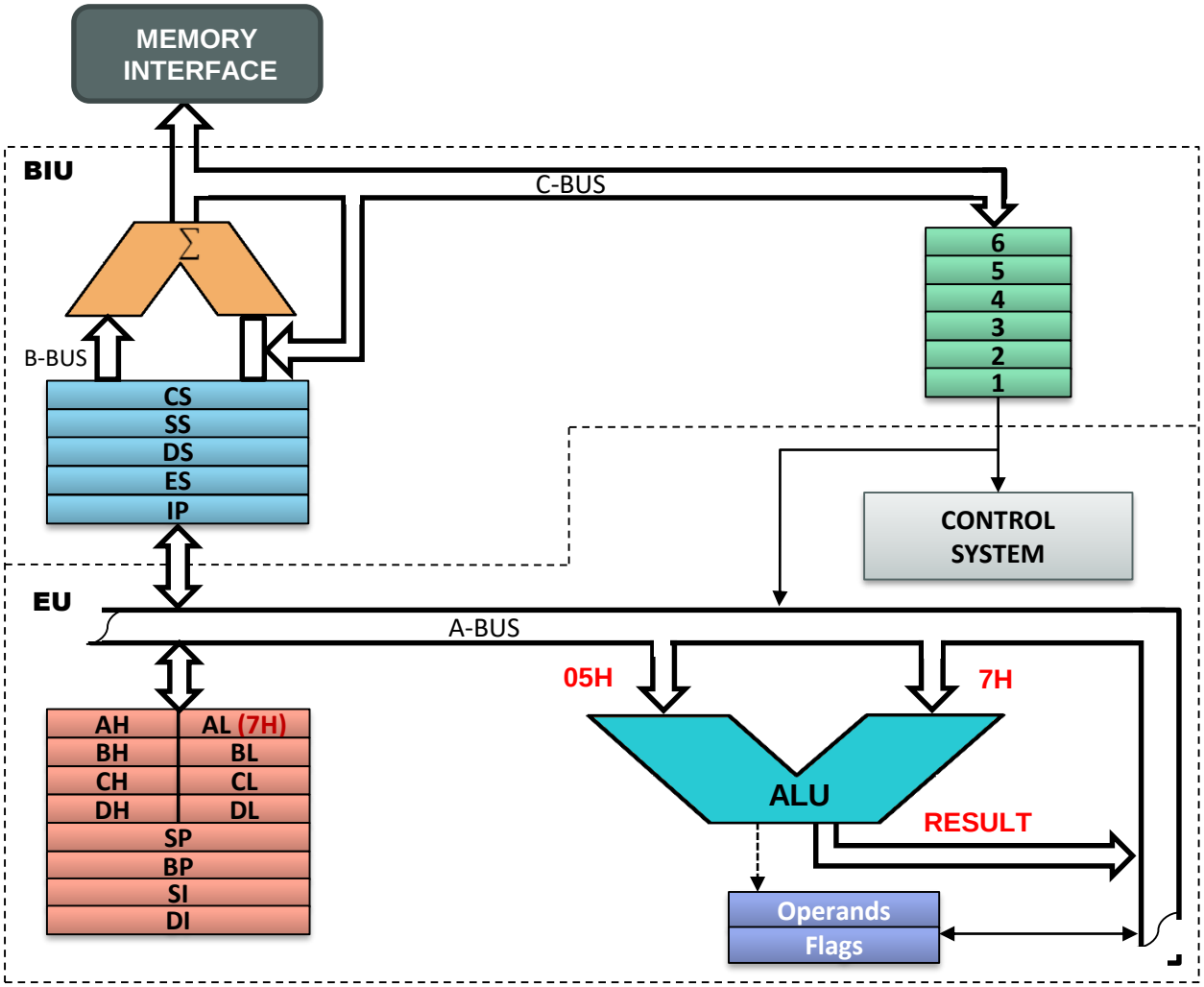
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

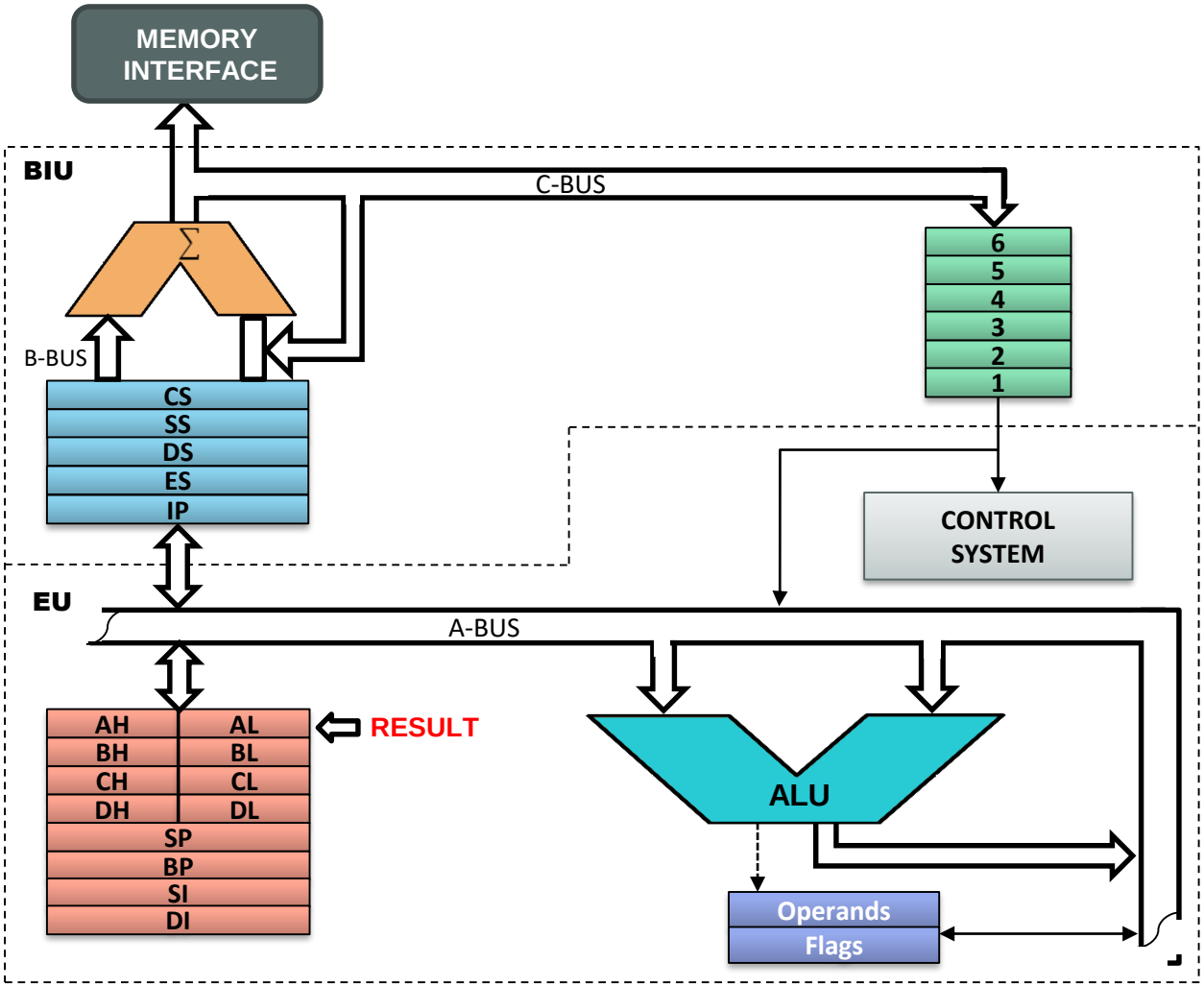
Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

Working of 8086 with an example



MOV AL, 7H

ADD AL, 05H

innovate achieve lead

BITS Pilani
Pilani Campus

Thank You