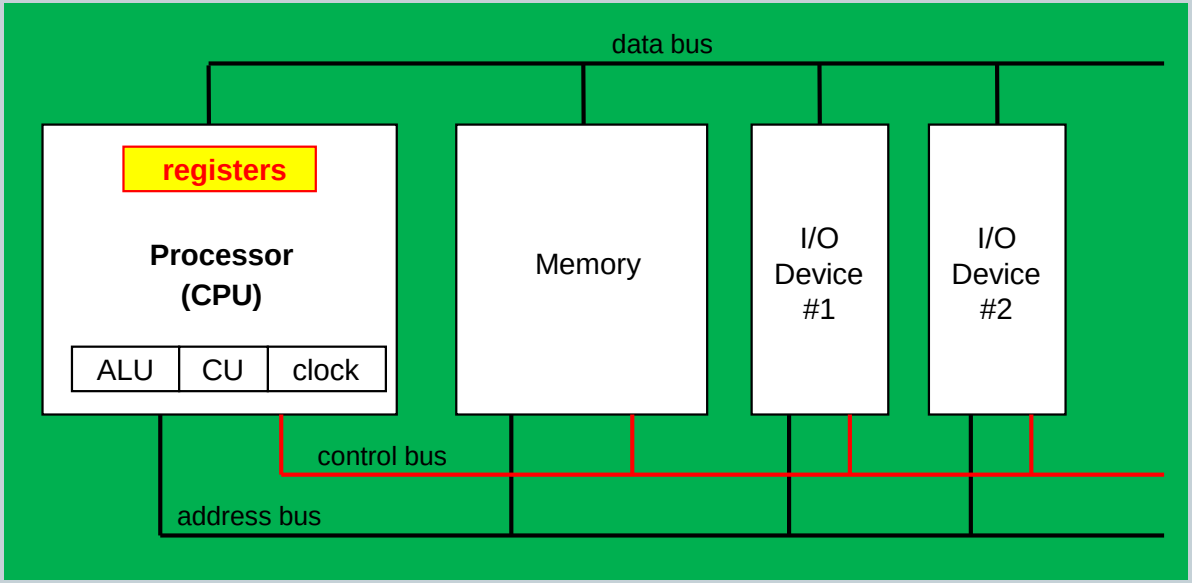
Instruction Set Architecture (ISA)

REGISTERS

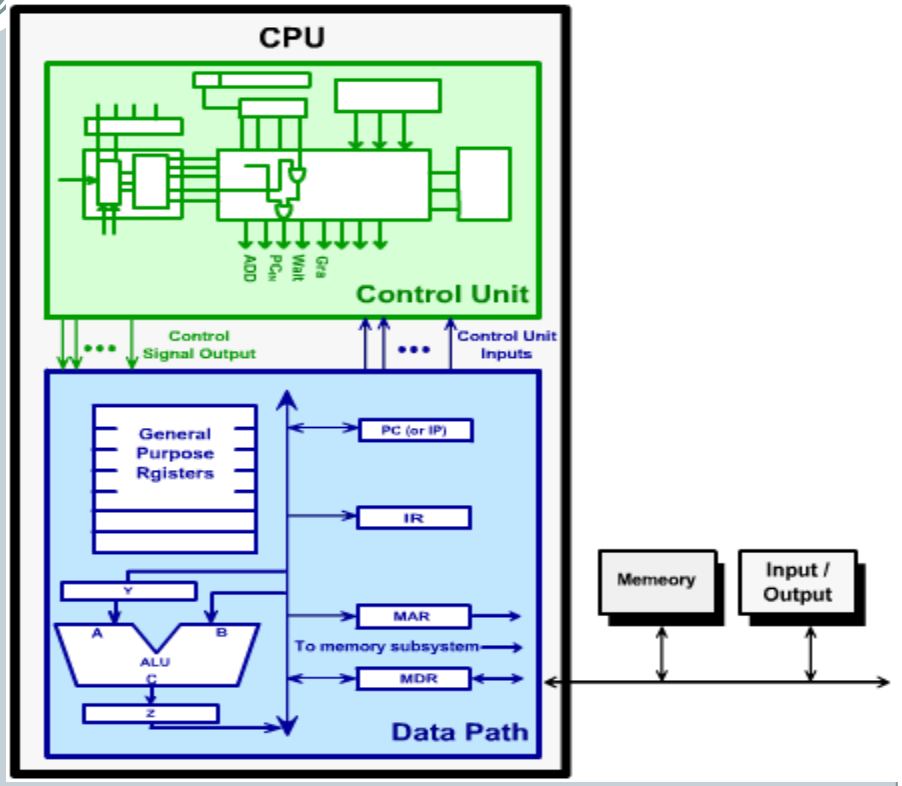
Basic Computer Organization

- Since the 1940's, computers have 3 classic components:
  - Processor, called also the CPU (Central Processing Unit)
  - Memory and Storage Devices
  - I/O Devices
- Interconnected with one or more buses
- Bus consists of
  - Data Bus
  - Address Bus
  - Control Bus



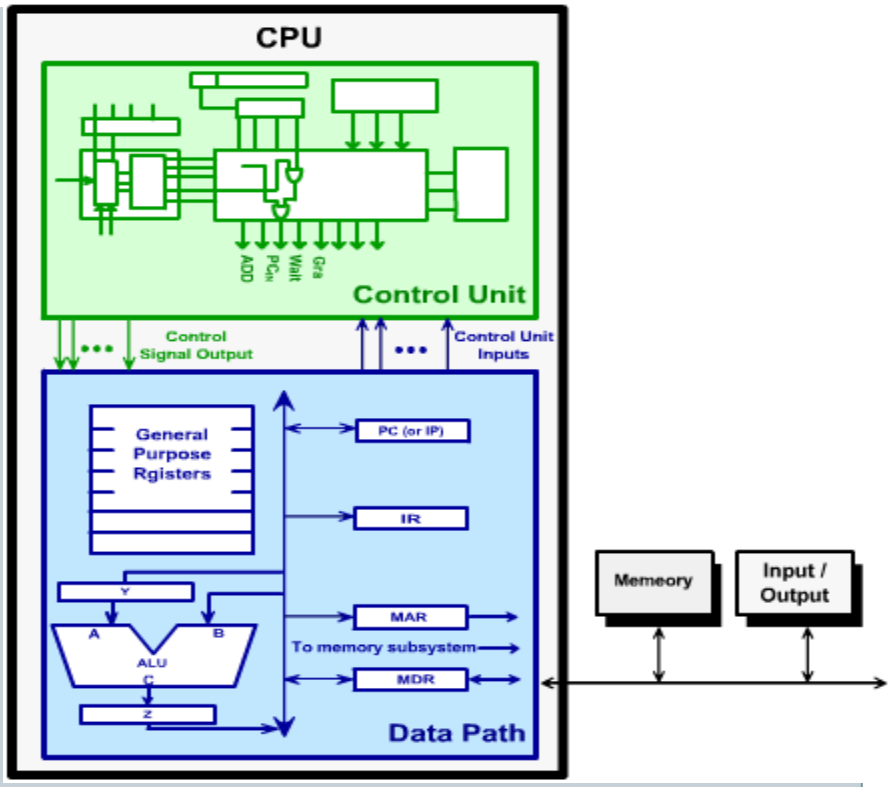
Processor (CPU)

- Processor consists of
  - Datapath
    - ALU
    - Registers
  - Control unit



Processor (CPU)

- ALU
  - Performs arithmetic and logic instructions
- Control unit (CU)
  - Generates the control signals required to execute instructions
- Implementation varies from one processor to another



Processor (CPU)

In performing its task, the processor (CPU) is partitioned into two logical units:

1) An Execution Unit (EU)
2) A Bus Interface Unit (BIU)

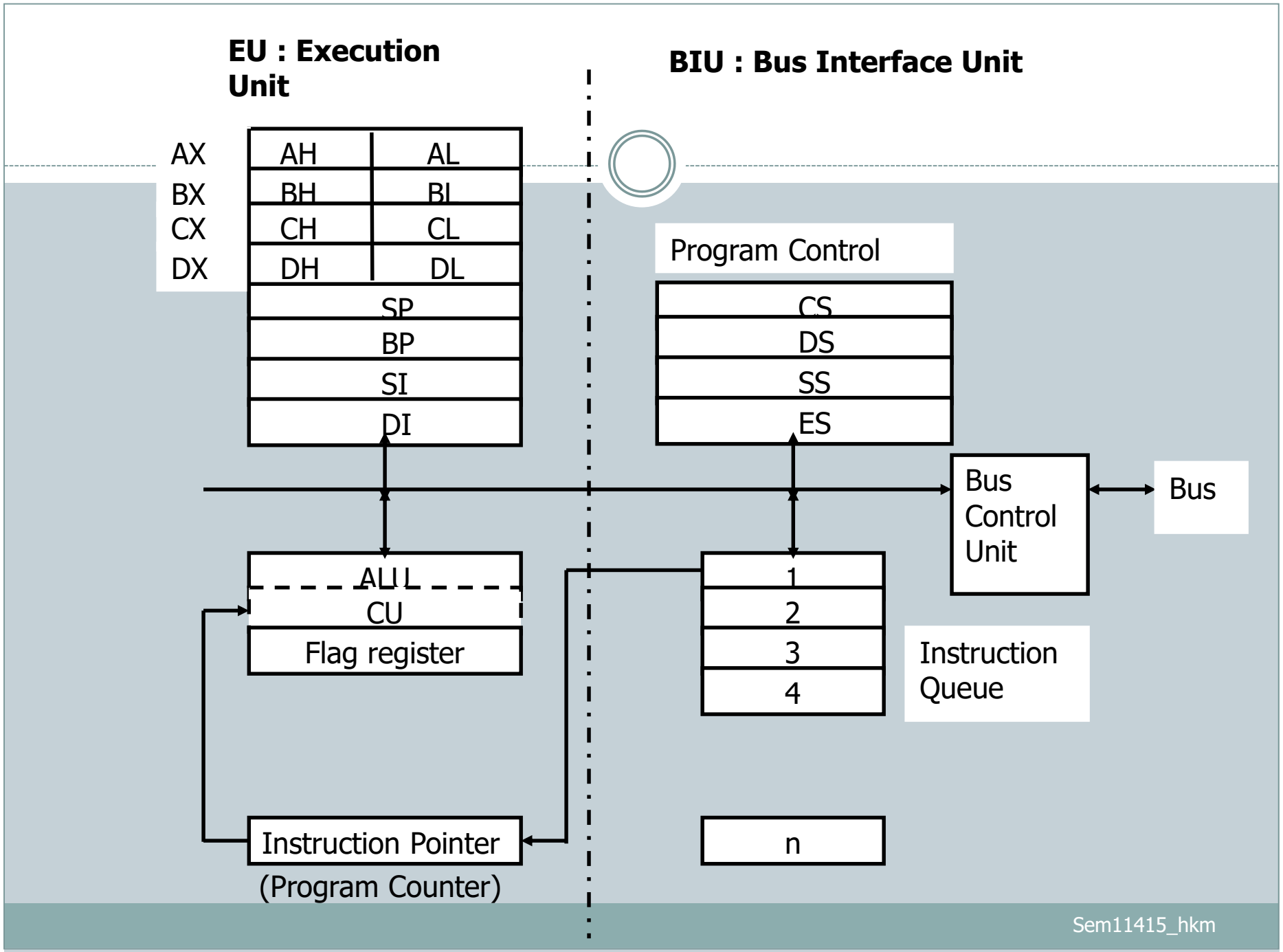
Processor (CPU)

EU

- EU is responsible for **program execution**
- Contains of an Arithmetic Logic Unit (ALU), a Control Unit (CU) and a number of registers

BIU

- **Delivers data and instructions** to the EU.
- manage the bus control unit, segment registers and instruction queue.
- The BIU controls the buses that transfer the data to the EU, to memory and to external input/output devices, whereas the segment registers control memory addressing.


EU : Execution Unit
BIU : Bus Interface Unit
AX
BX
CX
DX
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
Program Control
CS
DS
SS
ES
Bus Control Unit
Bus
ALU
CU
Flag register
1
2
3
4
Instruction Queue
Instruction Pointer
(Program Counter)
n

Processor (CPU)

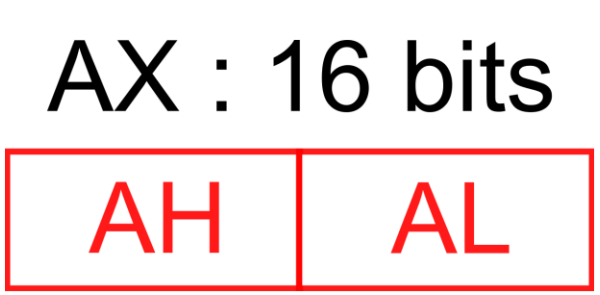
- EU and BIU work in parallel, with the BIU keeping one step ahead.
- The EU will notify the BIU when it needs
  - the data in memory **or**
  - an I/O device **or**
  - obtain instruction from the BIU instruction queue.
- When EU executes an instruction, BIU will fetch the next instruction from the memory and insert it into to instruction queue.

Traditional Registers in x86

- General Purpose Registers (GPR)
  - AX, BX, CX, DX
- Pseudo General Purpose Registers
  - Stack: SP (stack pointer), BP (base pointer)
  - Strings: SI (source index), DI (destination index)
- Special Purpose Registers
  - IP (instruction pointer) and EFLAGS

GPR usage

- Legacy structure: 16 bits
  - 8 bit components:
  - low and high bytes



- AX ← Accumulator (arithmetic)
- BX ← Base (memory addressing)
- CX ← Counter (loops)
- DX ← Data (data manipulation)

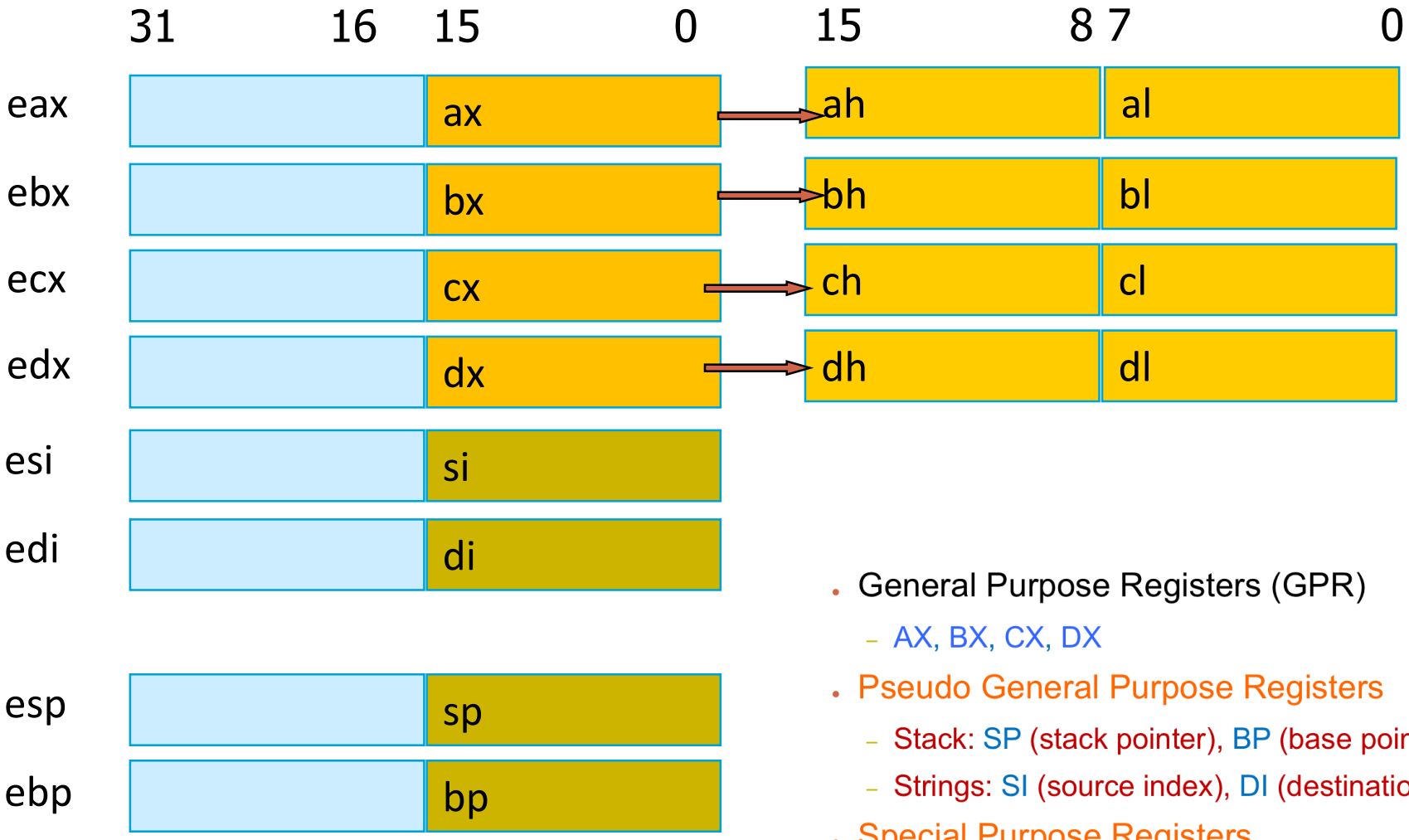
Registers (32-bit)

General purpose registers (GPR) ➔ primarily used for arithmetic and data movement

EAX	EAX ➔ Automatically used by MUL and DIV instructions
EBX	
ECX	ECX ➔ Automatically used by processor as a loop counter
EDX	
EBP	base pointer register
ESP	stack pointer register
ESI	source index register
EDI	destination index register

avoid usage for generic calculations!!!

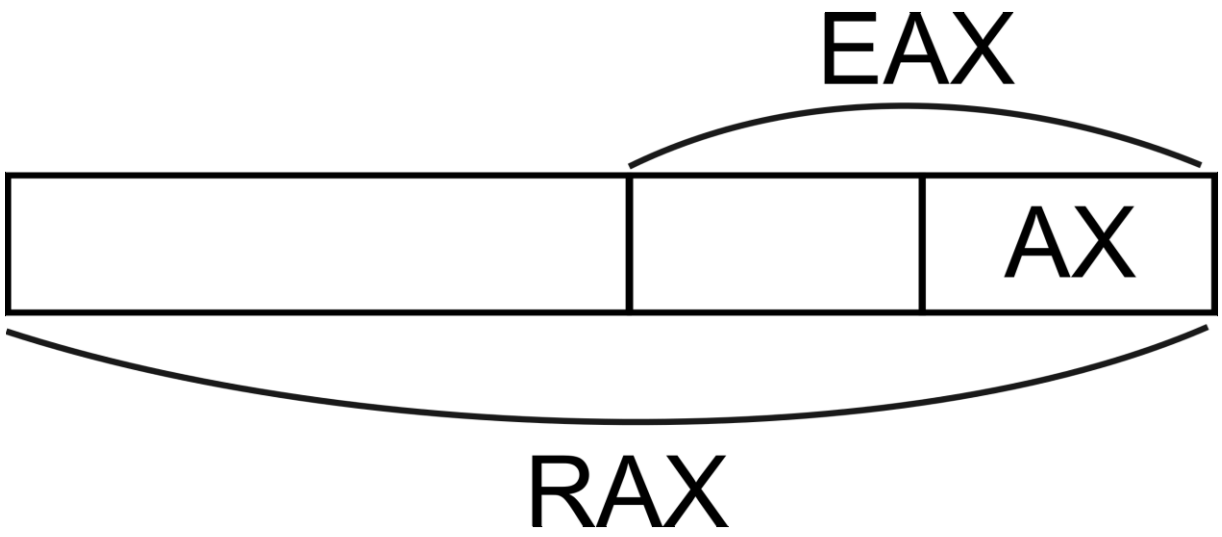
Registers (32-bit)



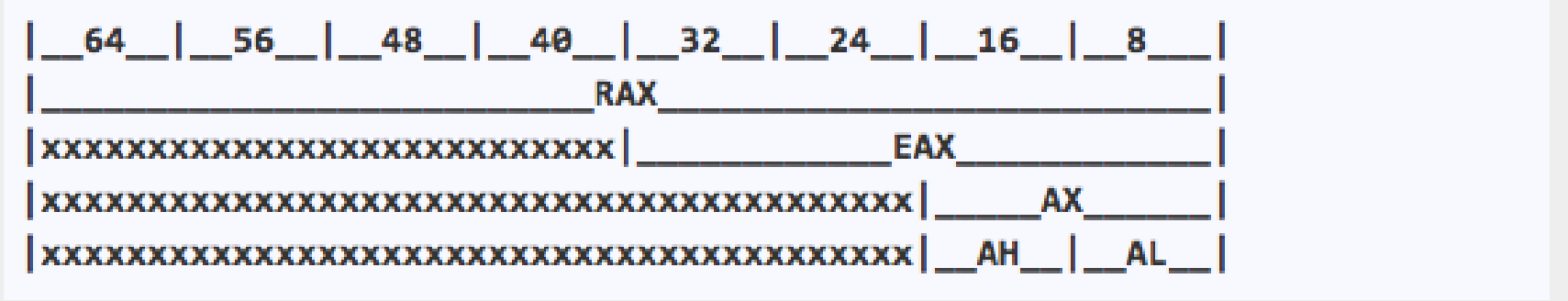
- General Purpose Registers (GPR)
  - AX, BX, CX, DX
- Pseudo General Purpose Registers
  - Stack: SP (stack pointer), BP (base pointer)
  - Strings: SI (source index), DI (destination index)
- Special Purpose Registers
  - IP (instruction pointer) and EFLAGS

Modern extensions (i.e. x64 arch)

- “E” prefix for 32 bit variants → **E**AX, **E**SP
- “R” prefix for 64 bit variants → **R**AX, **R**SP
- Additional GPRs in 64 bit: R8 →R15



64-bit Registers (current x64 arch)



x64 - GPR

64-bit register	Lower 32 bits	Lower 16 bits	Lower 8 bits
rax	eax	ax	al
rbx	ebx	bx	bl
rcx	ecx	cx	cl
rdx	edx	dx	dl
rsi	esi	si	sil
rdi	edi	di	dil
rbp	ebp	bp	bpl
rsp	esp	sp	spl

x64 – GPR (additional)

64-bit register	Lower 32 bits	Lower 16 bits	Lower 8 bits
r8	r8d	r8w	r8b
r9	r9d	r9w	r9b
r10	r10d	r10w	r10b
r11	r11d	r11w	r11b
r12	r12d	r12w	r12b
r13	r13d	r13w	r13b
r14	r14d	r14w	r14b
r15	r15d	r15w	r15b

Instruction Pointer Register

- The instruction pointer register (EIP) holds the address of the next instruction to be executed.
- The EIP register corresponds to the program counter (PC) register in other architectures.
- EIP can be manipulated for certain instructions (e.g. call, jmp, ret) to branch to a new location

Flags Register

- Earlier version (8086/8088) flags were 16 bits.
- Later versions flags are **32 bits** ➔ EFLAGS.
- Flags come in 2 types:
  - Conditional or status flags ➔
    - set or reset by the Execution unit (EU) on the basis of the results of some arithmetic operation.
  - Machine control flags ➔
    - used to control certain operations of processor.

Flags Register

- The EFLAGS register consists of individual binary bits that *control* CPU operation or reflect outcome (i.e. output or *status*) of some CPU operation.
- Some instruction test and manipulate individual processor flags.
  - Example :
    - STC – Set carry flag,
    - JNZ – Jump Not Zero

EFLAGS

31..22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ID	VIP	VIF	AC	VM	RF		NT	IOP1	IOP0	O	D	I	T	S	Z		A		P		C

8086808880186801888

80286

80386/8986DX

80486SX

PENTIUMPENTIUM 4

Flags Register : Status flags

- **carry flag (CF)**
  - indicates a carry after addition or a borrow after subtraction,
  - also indicates error conditions.
- **parity flag (PF)**
  - is a logic "0" for odd parity and a logic "1" for even parity.
- **auxiliary carry flag (AF)**
  - important for BCD addition and subtraction;
    - holds a carry (borrow) after addition (subtraction) between bits position 3 and 4.
  - ➔ *BCD is not used much anymore*
- **zero flag (ZF)**
  - indicates that the result of an arithmetic or logic operation is zero.

Flags Register : Status flags

- **sign flag (SF)**
  - indicates arithmetic sign of the result after an arithmetic operation.
- **overflow flag (OF)**
  - a condition that occurs when signed numbers are added or subtracted.
  - An overflow indicates that the result has exceeded the capacity of the machine.

Flags Register : Control flags

- used to control certain operations of processor.
- E.g.
  - can cause the CPU to break after every instruction executes,
  - interrupt when arithmetic overflow is detected,
  - enter virtual-8086 mode,
  - enter protected mode.

Extra knowledge ➔ you don't necessarily use this in this course

Flags Register : Control flags

- The control flags are deliberately set or reset with specific instructions YOU put in your program.
- **trap flag (TF) -** used for single stepping through a program;
- **interrupt flag (IF)** - used to allow or prohibit the interruption of a program;
- **direction flag (DF) -** used with string instructions.

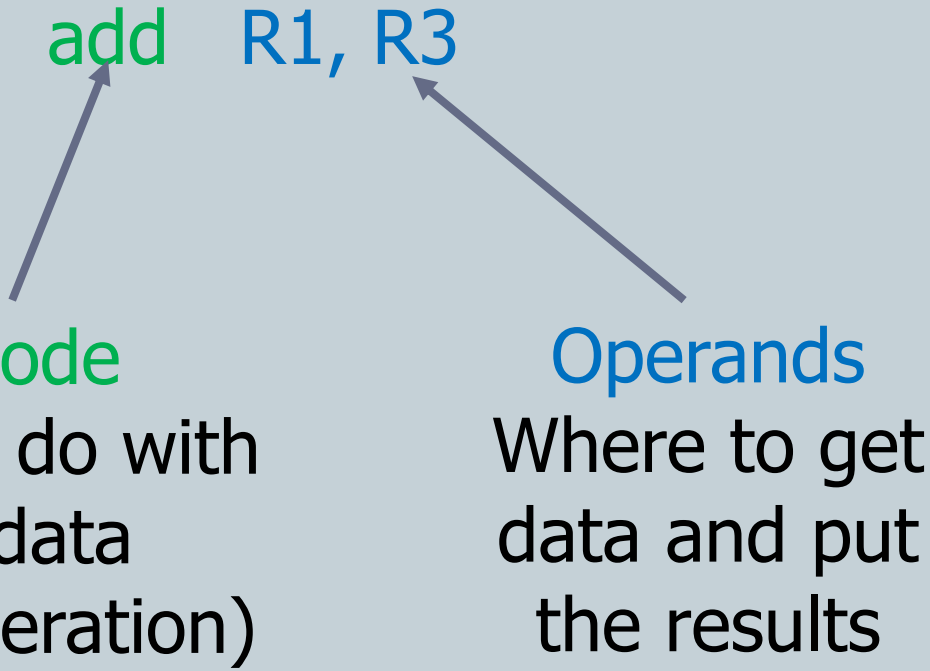
Extra knowledge ➔ you don't necessarily use this in this course

Instruction Set Architecture (ISA)

TYPES OF OPERAND

Types of Operand

- Assembly language built from two pieces:



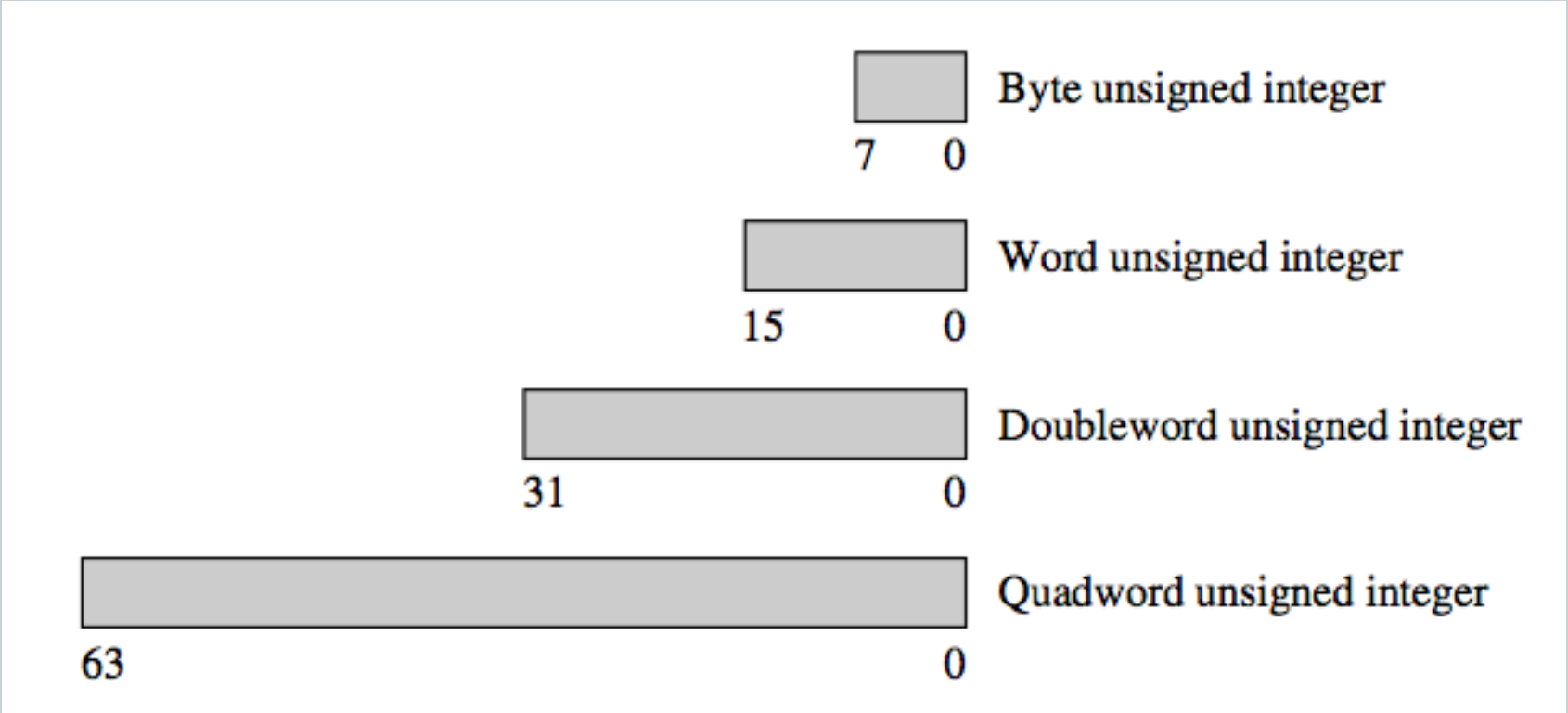
Types of Operand

- Machine instruction operate on data.
- Most important general categories of data are:
  - Numbers – numeric data
    - Integer/floating point/decimal
    - Limited magnitude of numbers – integer/decimal
    - Limit precision – floating point
  - Characters – data for text and strings
    - ASCII, UNICODE etc.
  - Logical Data
    - Bits or flags

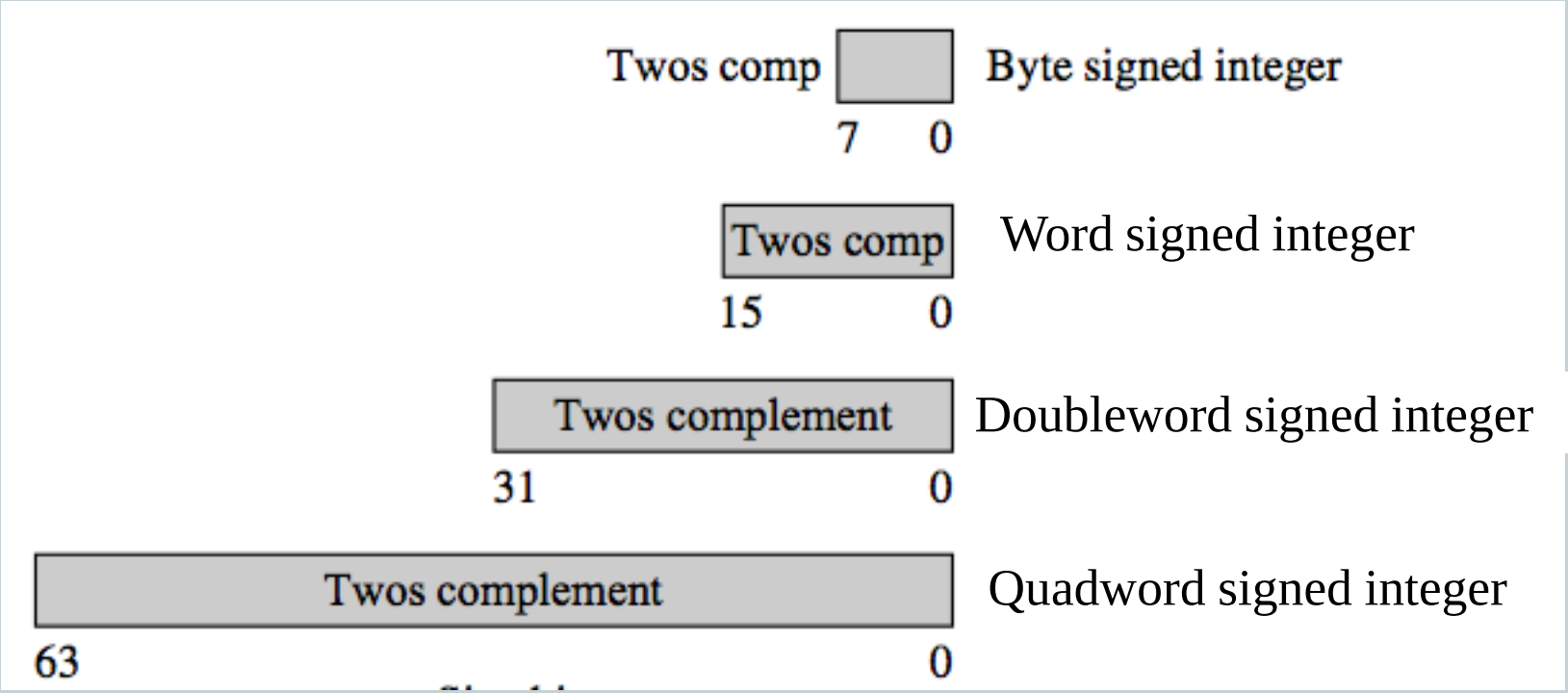
Example: Types of Operand for Pentium 4

Type	1 Bit	8 Bits	16 Bits	32 Bits	64 Bits	128 Bits
Bit						
Signed integer		X	X	X		
Unsigned integer		X	X	X		
Binary coded decimal integer		X				
Floating point				X	X	

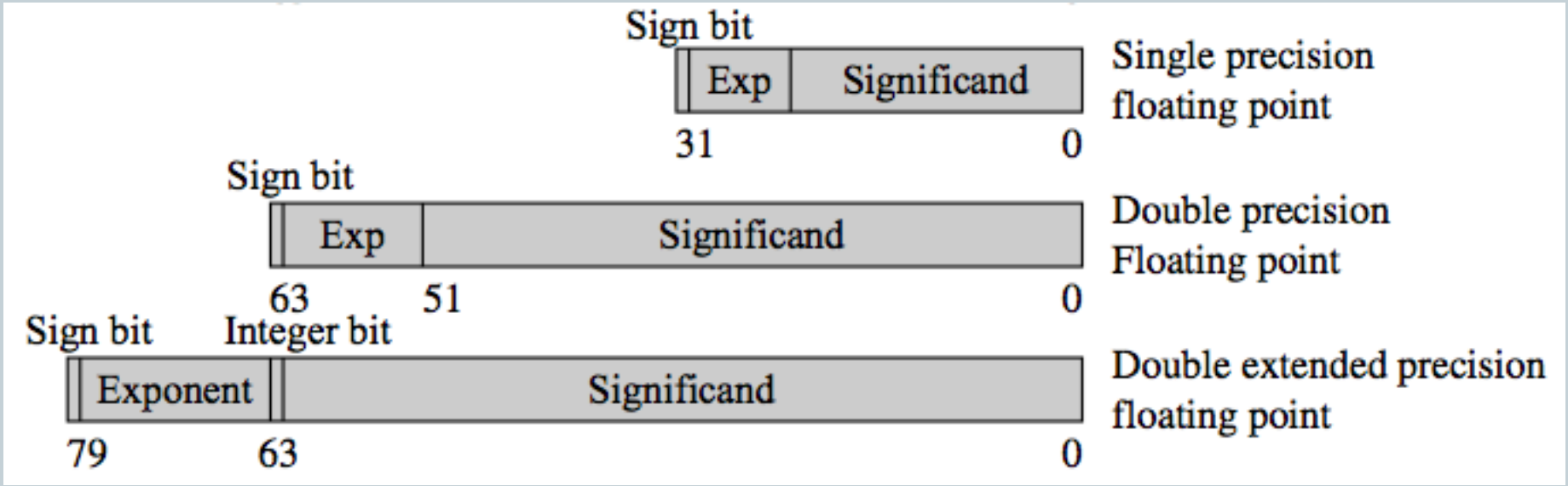
x86 Numeric Data Formats



x86 Numeric Data Formats



x86 Numeric Data Formats



Type	Size in bits	Format	Value range	
			Approximate	Exact
character	8	signed (one's complement)	-127 to 127	
		signed (two's complement)	-128 to 127	
		unsigned	0 to 255	
integral	16	signed (one's complement)	$\pm 3.27 \cdot 10^4$	-32767 to 32767
		signed (two's complement)		-32768 to 32767
		unsigned	0 to $6.55 \cdot 10^4$	0 to 65535
	32	signed (one's complement)	$\pm 2.14 \cdot 10^9$	-2,147,483,647 to 2,147,483,647
		signed (two's complement)		-2,147,483,648 to 2,147,483,647
		unsigned	0 to $4.29 \cdot 10^9$	0 to 4,294,967,295
	64	signed (one's complement)	$\pm 9.22 \cdot 10^{18}$	-9,223,372,036,854,775,807 to 9,223,372,036,854,775,807
		signed (two's complement)		-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
		unsigned	0 to $1.84 \cdot 10^{19}$	0 to 18,446,744,073,709,551,615
floating point	32	IEEE-754	$\pm 3.4 \cdot 10^{\pm 38}$ (~7 digits)	• min subnormal: $\pm 1.401,298,4 \cdot 10^{-47}$ • min normal: $\pm 1.175,494,3 \cdot 10^{-38}$ • max: $\pm 3.402,823,4 \cdot 10^{38}$
	64	IEEE-754	$\pm 1.7 \cdot 10^{\pm 308}$ (~15 digits)	• min subnormal: $\pm 4.940,656,458,412 \cdot 10^{-324}$ • min normal: $\pm 2.225,073,858,507,201,4 \cdot 10^{-308}$ • max: $\pm 1.797,693,134,862,315,7 \cdot 10^{308}$

… end of Part 3

- Hierarchy of Computer Languages
- General Concepts
- ISA Level
- Elements of Instructions
- Instructions Types
- Instruction Formats
- Number of Addresses
- **Registers**
- **Types of Operands**
- Addressing Modes

Part 3