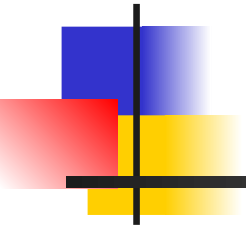


Computer Architecture, Networks, and Operating Systems (CANOS)

Lecture 13:
x86 (part 1)

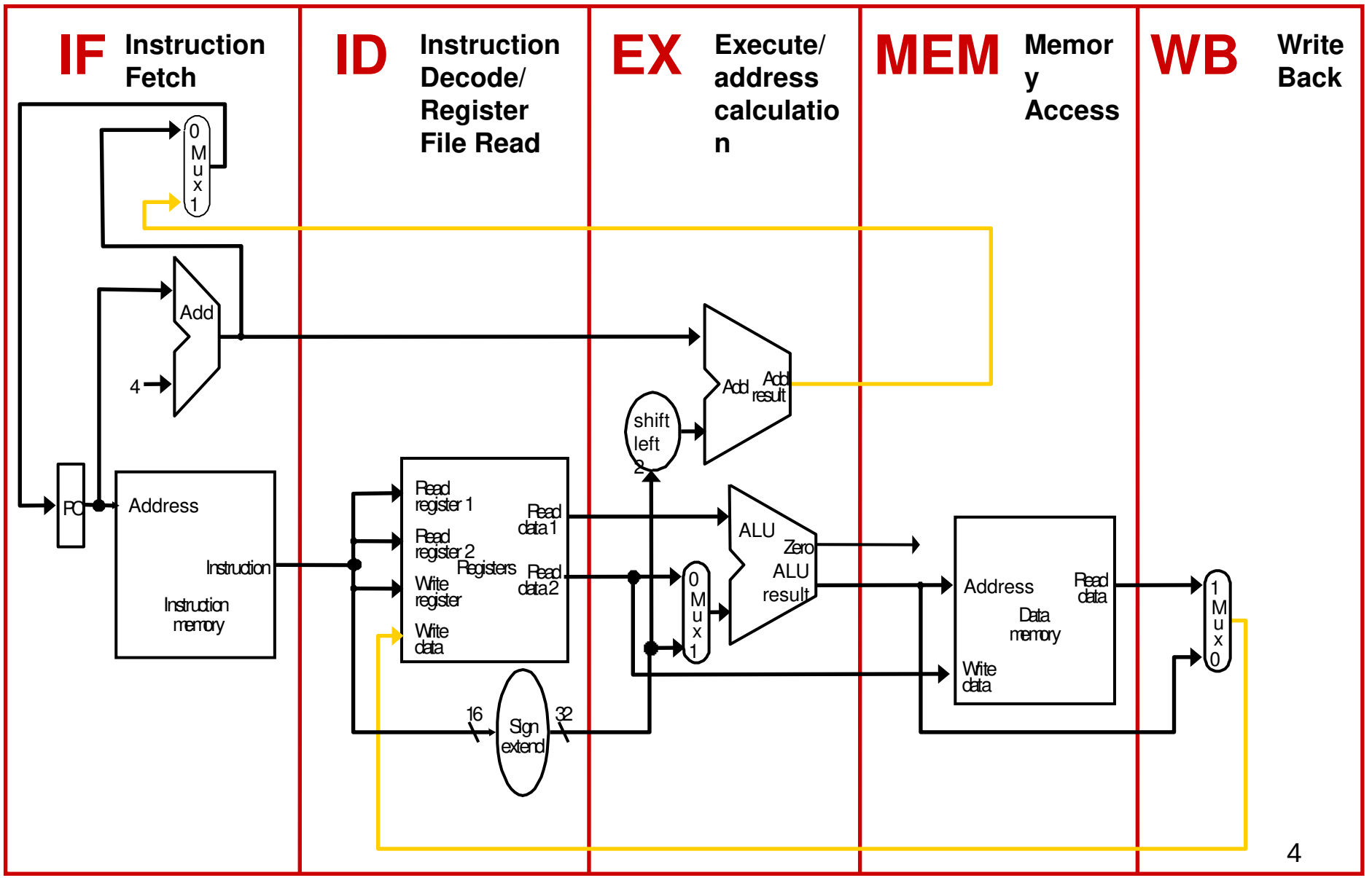
Announcements

- Exam 1 results: **out ASAP**
- Will be out of town next Friday. You will have a remote lecture
- Homework 5 due Tuesday
- Student presentation

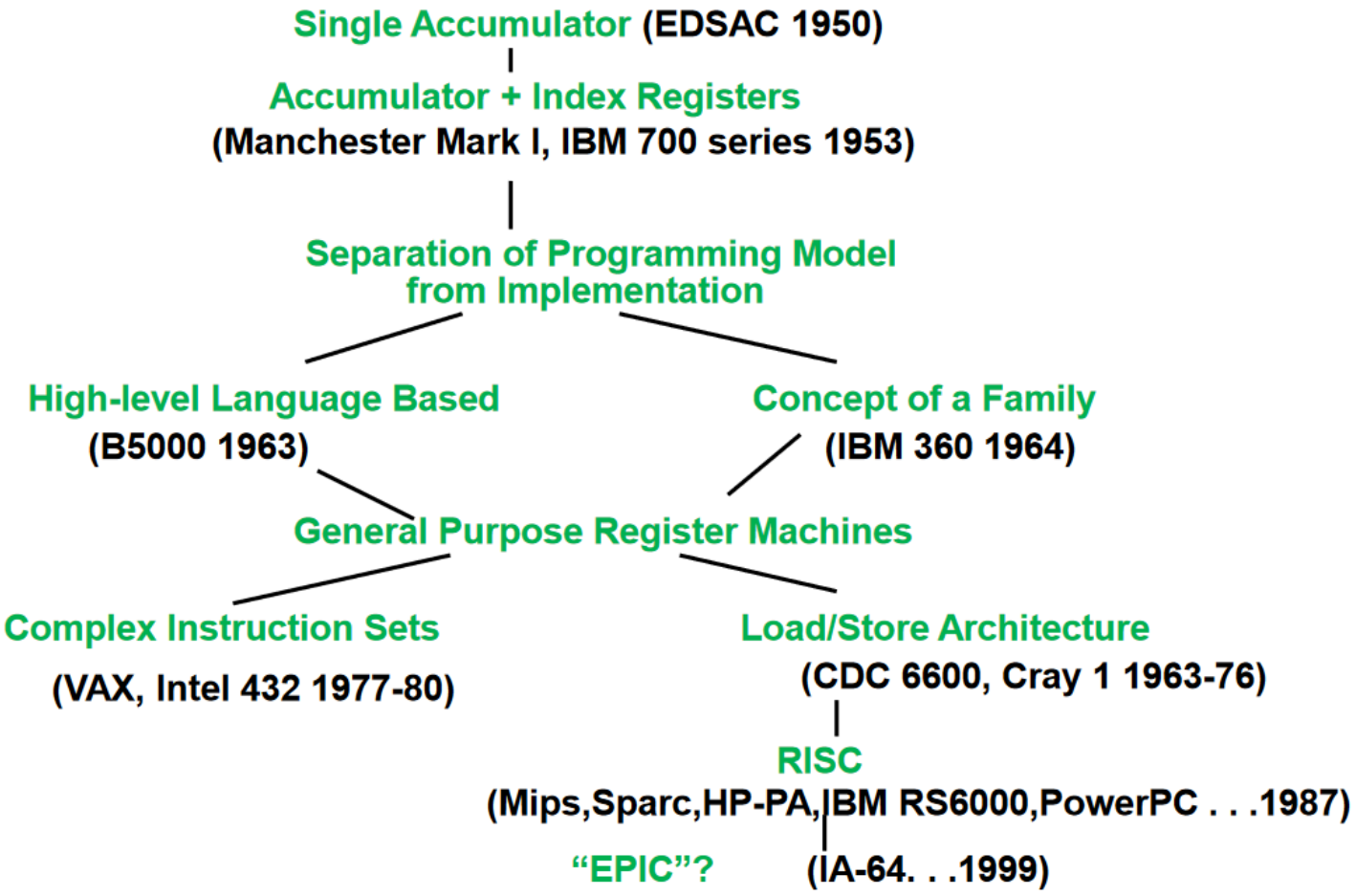
Metacognition

- Control hazards review w/ pipeline diagram

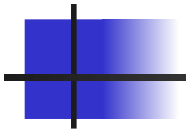
```
beq $s2, $s3, Label
sub $t0, $s0, $s1
sub $t0, $s0, $s1
sub $t0, $s0, $s1
sub $t0, $s0, $s1
j End
Label:
add $t0, $s0, $s1
add $t0, $s0, $s1
add $t0, $s0, $s1
add $t0, $s0, $s1
End:
```


IF
Instruction Fetch
ID
Instruction Decode/ Register File Read
EX
Execute/ address calculation
MEM
Memory Access
WB
Write Back
0
M
u
x
1
Add
4
PC
Address
Instruction
Instruction memory
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
shift left 2
Add
Add result
0
M
u
x
1
ALU
Zero
ALU result
Address
Data memory
Read data
Write data
1
M
u
x
0

Evolution of Instruction Sets



Source: Dreslinsky (2022)

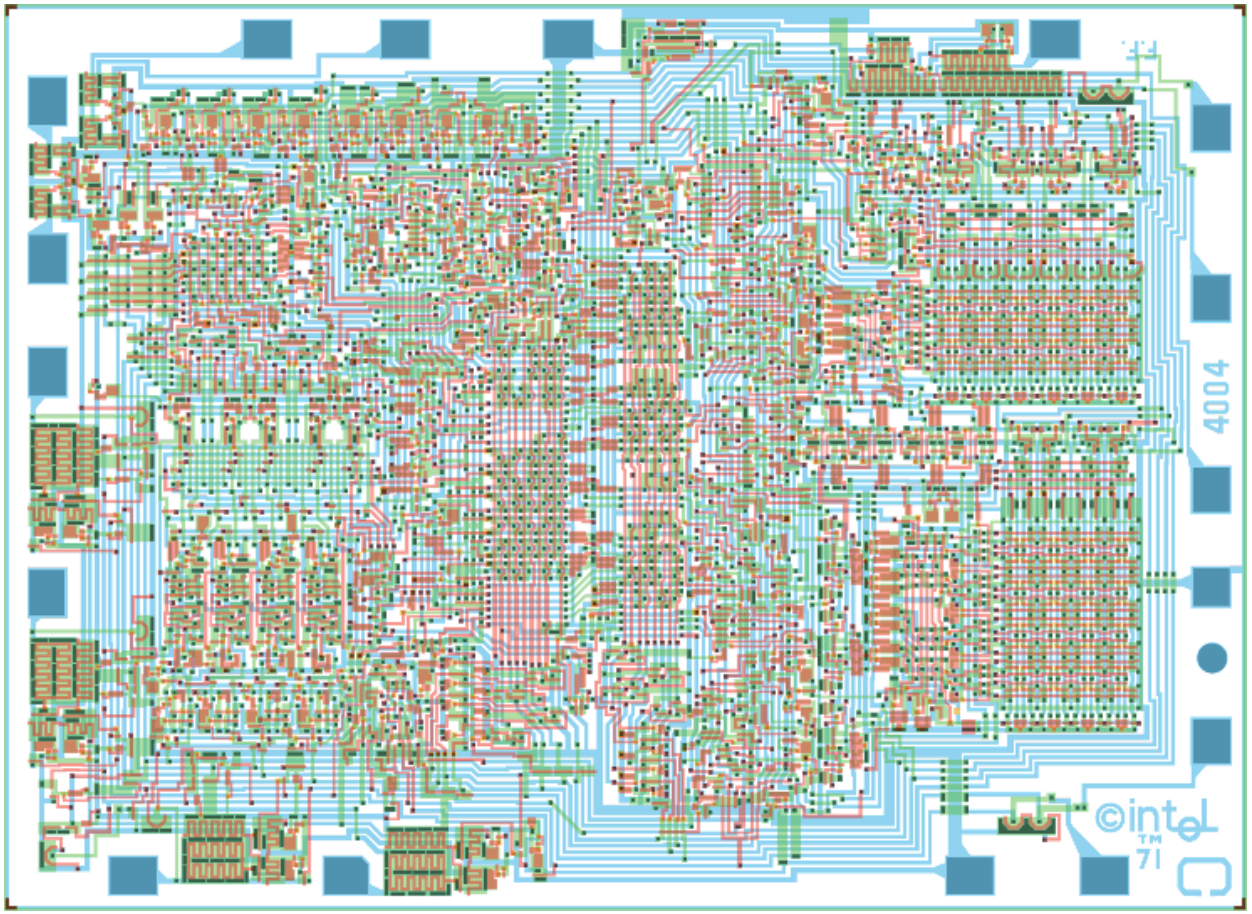


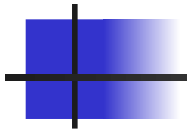
CISC vs RISC

- x86 machines are CISC (complex instruction set architecture) to be contrasted with MIPS which is RISC (reduced instruction set architecture)
- This generally means that x86 instructions are more powerful, more varied, and more difficult to implement in hardware
- More powerful instructions = fewer lines of code = less CPU time used (sometimes)
- It also means more potential for pipeline hazards

x86 History

- Intel 4004 (1971) – first commercial microprocessor. 4 bit data width, 12 bit address width

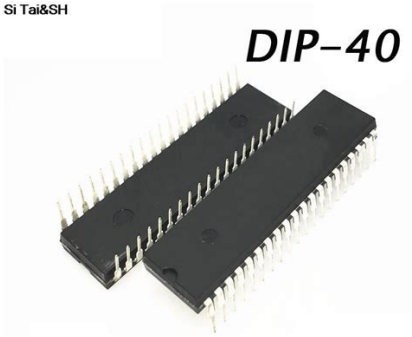




x86 History

- Intel 8008 (1972)
  - Based on the 4004
  - Faster clock rate (200 kHz to 800 kHz)
  - Data width increased to 8 bits
  - Expended instruction set
  - More transistors

x86 History



- Intel 8080 (1974)
  - Based on the 8008
  - Faster clock rate (2 MHz)
  - 16-bit stack pointer and 16-bit program counter greatly increased ease of addressing memory and performing program control logic
  - DIP40 socket
  - Much more useful as a general-purpose computer than the 4004 or 8008; helped kickstart the home computer industry

x86 History

- Intel 8086 (1978)
  - Based on the 8080
  - 16-bit data width, 20-bit address width (8080 was 8-bit)
  - Faster clock rate (5 MHz – 10 MHz)
  - Was marked by Intel for desktop computers and helped create this market
  - The x86 architecture family begins here (modern x86 computers are backward compatible with the 8086!)

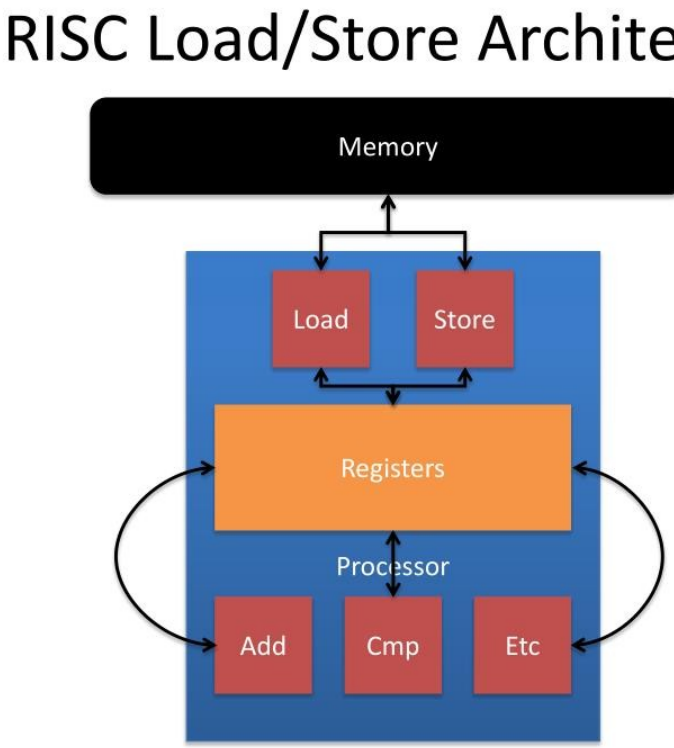
Table 0.1 | Year of Release of Various x86 Processors

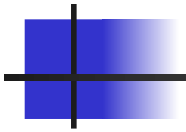
x86 processor	Year of release
8086	1978
8088	1979
80186	1982
80286	1982
89386	1985
80486	1989
Pentium	1993
Pentium Pro	1995
Pentium-2	1997
Pentium-3	1999
Pentium-M	2002
Pentium-4	2004
Pentium-D	2005
Core-2	2006
Core-2 Quad	2007

Source: Das

x86 / MIPS Comparison

- MIPS is a load-store architecture
- This means that load and store are the only instructions that interact with memory (and they do so using a very specific instruction fo



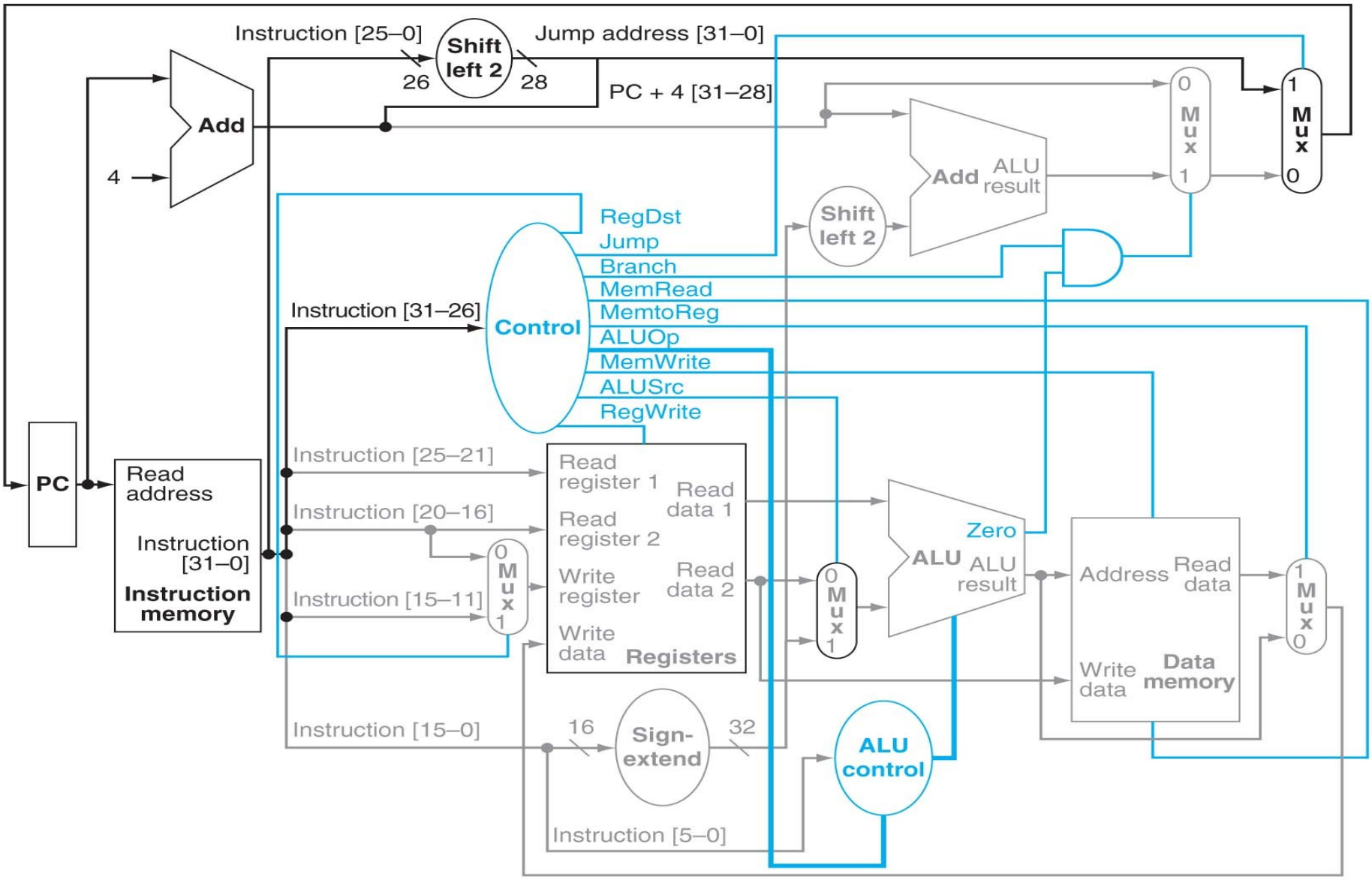


x86 / MIPS Comparison

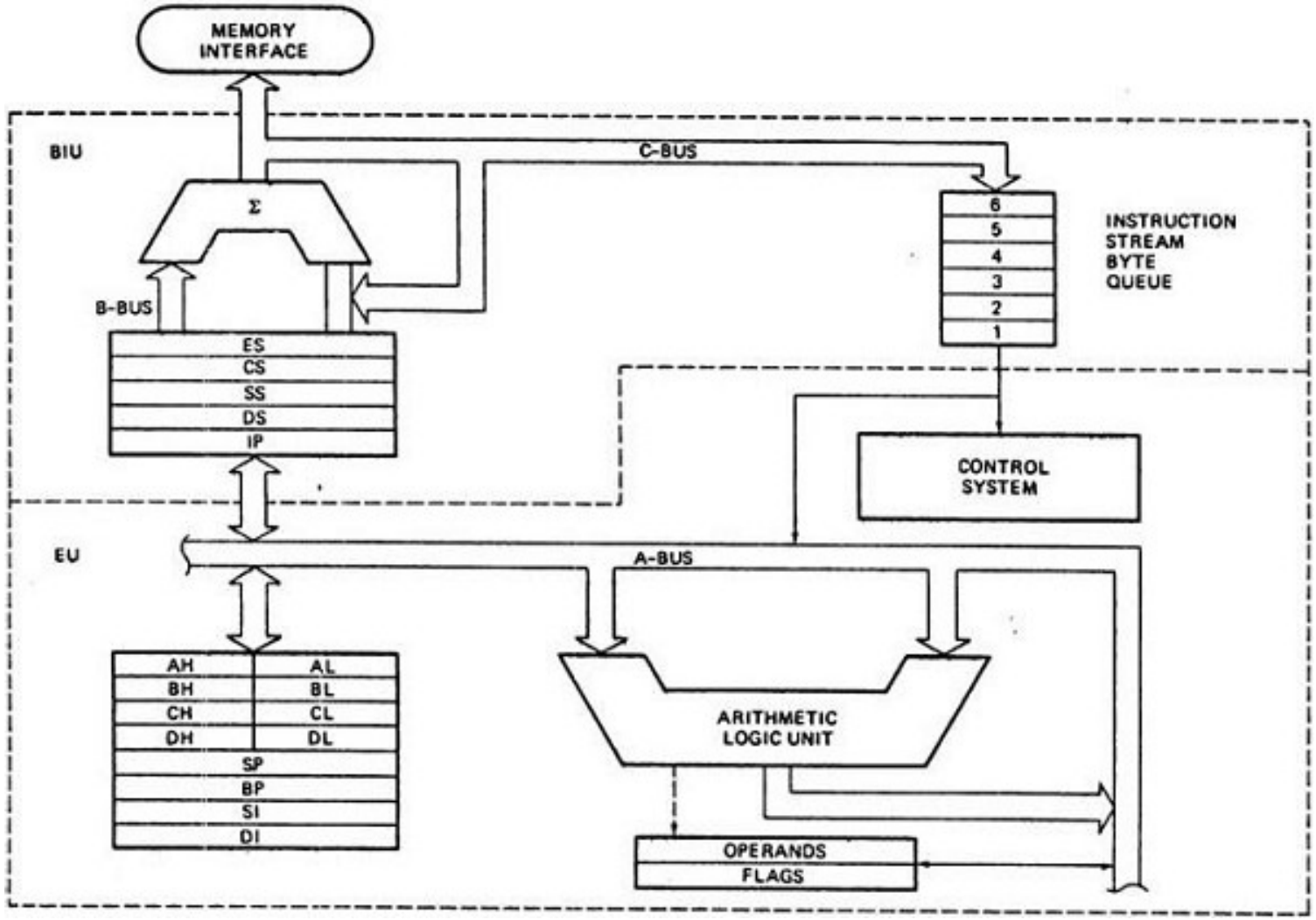
- x86 is a register-memory architecture
- It has fewer formatting restriction on interaction with memory; for instance, arithmetic may use two operands as registers or a register and a memory location (but not two memory locations)
- Example:

```
add [eax], ecx
```

MIPS High-Level Architecture



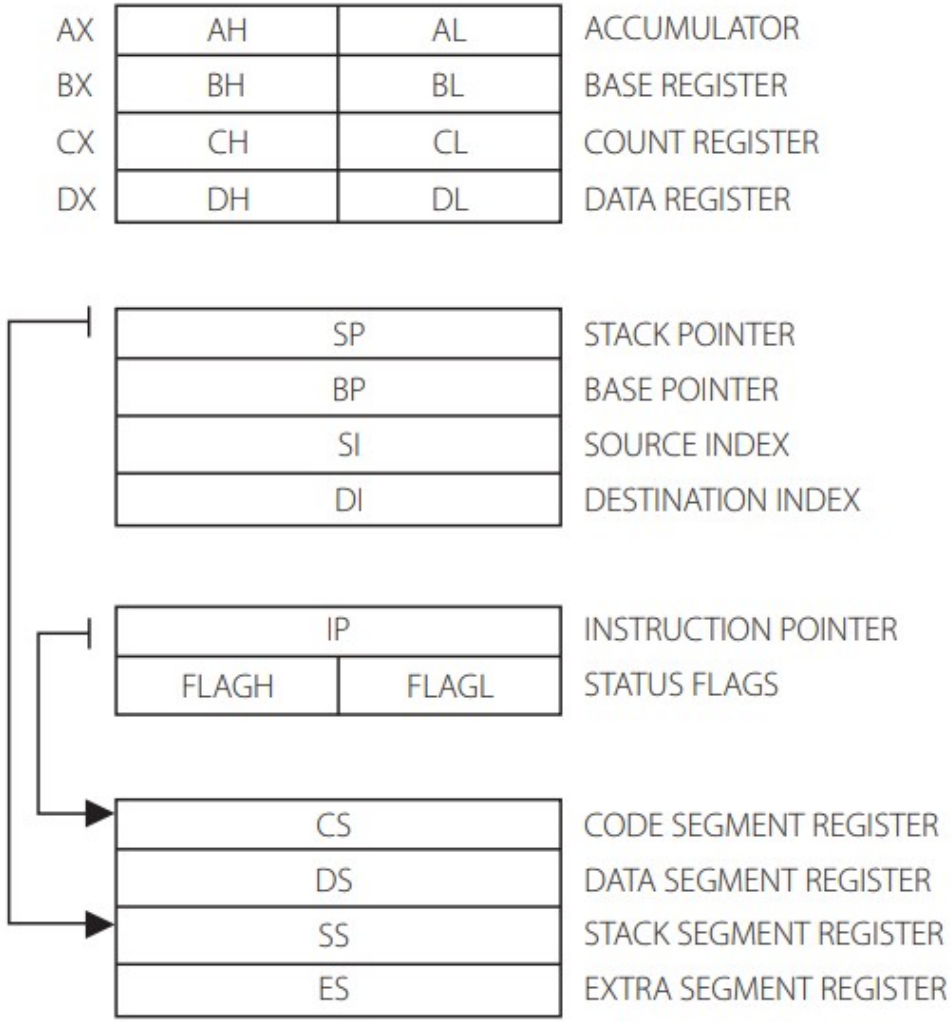
8086 High-Level Architecture



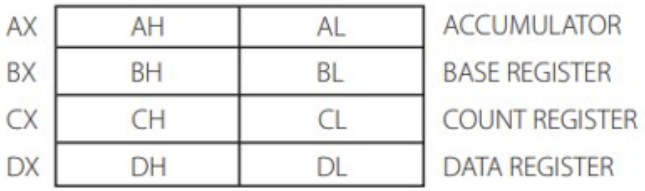
MIPS Registers

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$kl	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	Yes

8086 Registers



8086 Scratchpad Registers

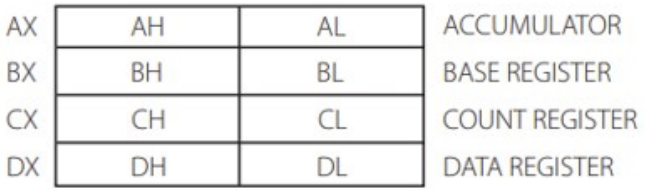


- AX, BX, CX, DX are general purpose 16-bit registers
- Each one can also be used as two 8 bit registers. AX contains AH and AL, BX contains BH and BL, etc.
- These 4 registers have semi specialized purposes (Accumulator, base address, etc.) but are still general purpose.
- Example:

```
mov ah, 5h
mov al, 5h
mov ax, 5h
```

```
mov ax, 555h
mov al, 555h
```

8086 Scratchpad Registers



- SP, BP, SI, DI are address registers. 16 bit only
- BP, SP = base pointer and stack pointer. Sp points to top of stack, bp can point to anywhere in the stack.
- SI, DI = source index and destination index. Used for strings

8086 Instruction Pointer + Flag Registers

- Instruction pointer = program counter
- Flag register = 16-bit register, of which 7 bits are unused.
- Remaining 6 bits are flags
  - Carry (CF) – set if there is a carry out from MSB during arithmetic
  - Zero (ZF) – set when an arithmetic result is zero
  - Parity (PF) – even parity in LSB of destination
  - Overflow (OF) – indicates overflow of MSB into sign bit
  - Sign Flag (SF) – set if sign bit is 1 (negative result)
  - Auxiliary Carry Flag (AF) – carry out of lower 4 bits (D3 to D4)

8086 Flag Register

Example 1.1

Find the status of the flags CF, SF, AF after the following instructions are executed.

```
MOV AL, 35H
ADD AL, 0CEH
```

Solution

The format of a MOV instruction is MOV destination, source.
In the above two instructions, a number 35H is first moved to AL. In the next line, 0CEH is added to AL. The sum will be in AL, the destination register.

```
   35H        0011 0101   +
+  CEH        1100 1110
  103H      1 0000 0011
```

After computation, AL will contain 0000 0011 and
CF = 1 since there is a carry out from D7.
SF = 0 since the sign bit (MSB) of the 8-bit destination is 0.
AF = 1 since there is an overflow from D3 to D4.

Source: Das

8086 Flag Register

Example 1.2

Show the effect of the following instructions on the CF, ZF and OF bits of the flag register.

```
MOV BX, 45ECH
ADD BX, 7723H
```

Solution

This is the case of 16-bit addition

	45ECH	0100 0101 1110 1100
+	7723H	0111 0111 0010 0011
	BD0FH	**1011 1101 0000 1111**

The sum will be in BX, which is the destination register.
CF = 0 since there is no carry out from D15.
ZF = 0 since the destination is not zero.
OF = 1 since there is an overflow into the MSB D15.

urce: Das

8086 Segment Registers

- 16 bits. Each one stores the base address of the corresponding memory segment
- CS (code segment register): stores the base location of the code section (.text section) which is used for data access
- DS (data segment register): Data segment register stores the default location for variables (.data section) which is used for data access.
- SS (stack segment register): Stack segment register stores the base location of the stack segment and is used when implicitly using the stack pointer or when explicitly using the base pointer.
- ES (extra segment register)

8086 Memory

- 20-bit address bus
- Memory is divided into 4 different segments labeled as data, code, stack and extra. This is called memory segmentation and allows code and data to be kept separate
- To generate the 20-bit address, the data segment register is shifted left 4 bits then added to the appropriate segment register

```
mov [200h], 5h
```

8086 Addressing Modes

- Register addressing – both source and destination are registers

  MOV AL, AH

- Immediate addressing – source is constant data

  MOV AL, 45H

- Direct addressing – either source or destination is a memory address

  MOV AX, [2345H]

  MOV [1089H, AL]

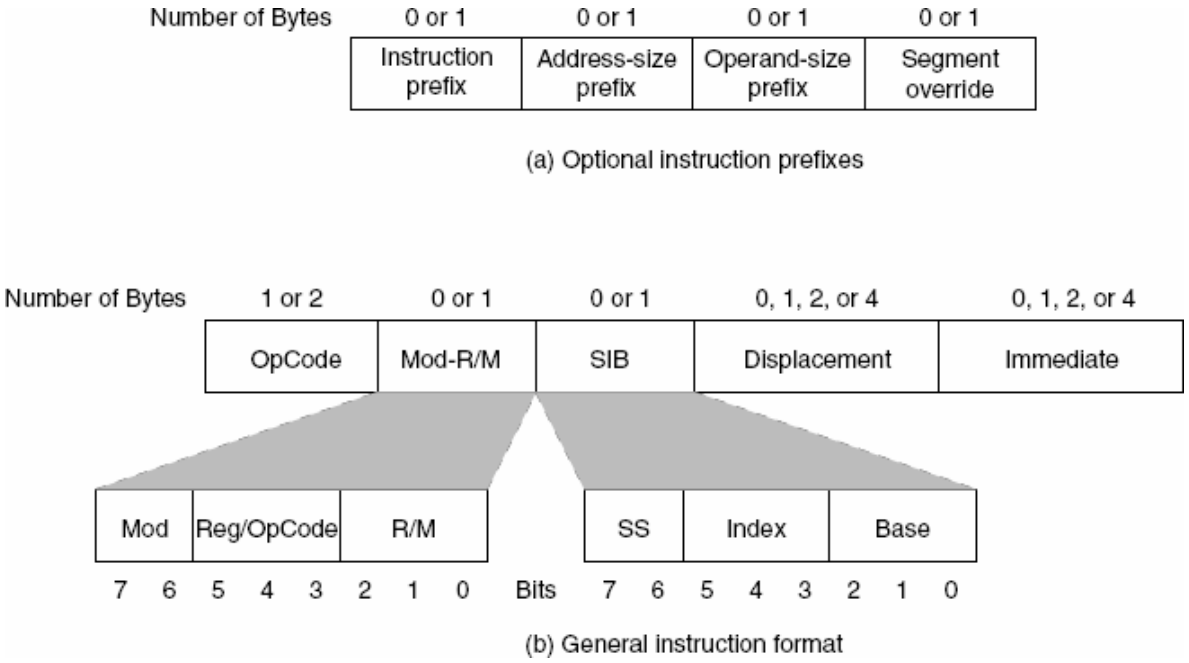
8086 Addressing Modes

- Register indirect addressing – uses a register as a pointer

MOV AL, [BX]

8086 Instructions

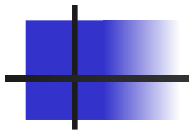
- Length varies – 1-6 bytes
- 2 addresses instead of 3 like mips
- Add a,b means a = a+b



8086 Instruction Example

mov ax, 23H

MOV [31H], ax



8086 Instruction Example

```
mov [200h], 5h
mov cx, [200h]
mov ax, 1h
mov dx, 0h
mult:
mul cx
loop mult
mov [300h], ax
mov [301h], dx
```

Exercise

- Download Emu8086
- https://emu8086.en.lo4d.com/windows
- Write pseudocode for each line of Example 1.
- Use the following reference to help you:
- https://eecs.wsu.edu/~ee314/handouts/x86ref.pdf

Wrap-Up

- Reading:
  - P+H 5.5, 5.10
  - P+H 2.16 (“Real Stuff: x86”)
  - Das chapter 1 (on shared drive)