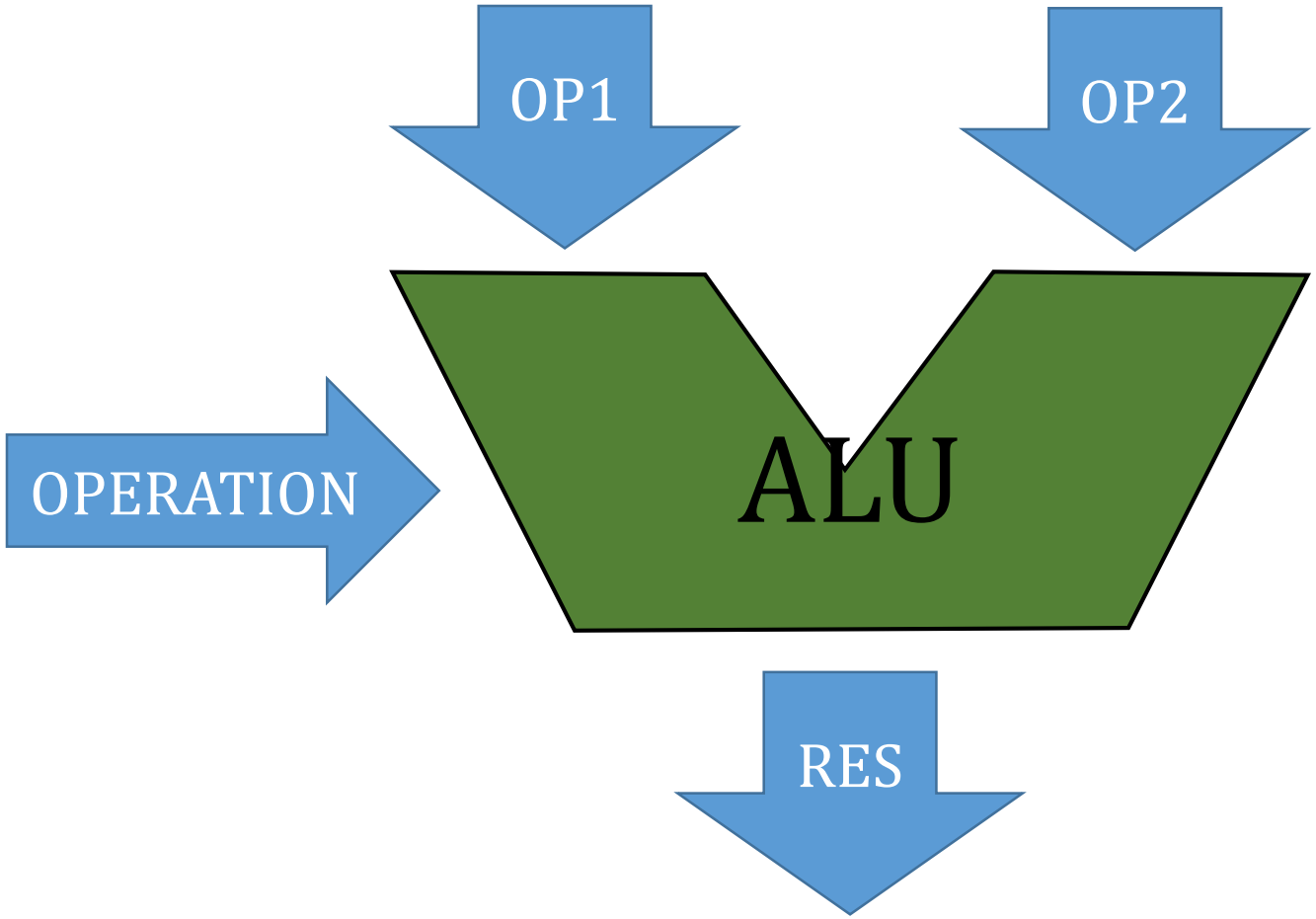
Instruction Set Architectures

Computer Systems: Section 4.1

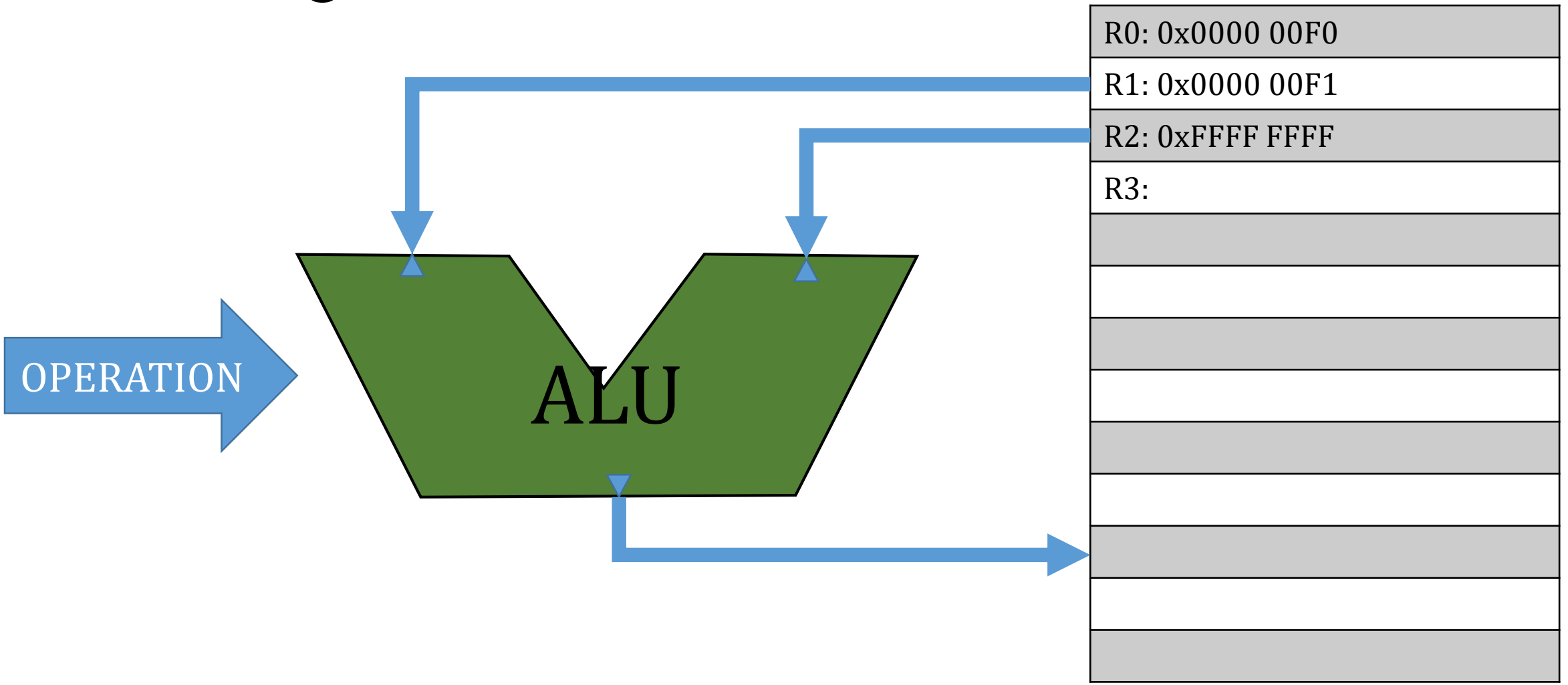
Suppose you built a computer…

What Building Blocks would you use?

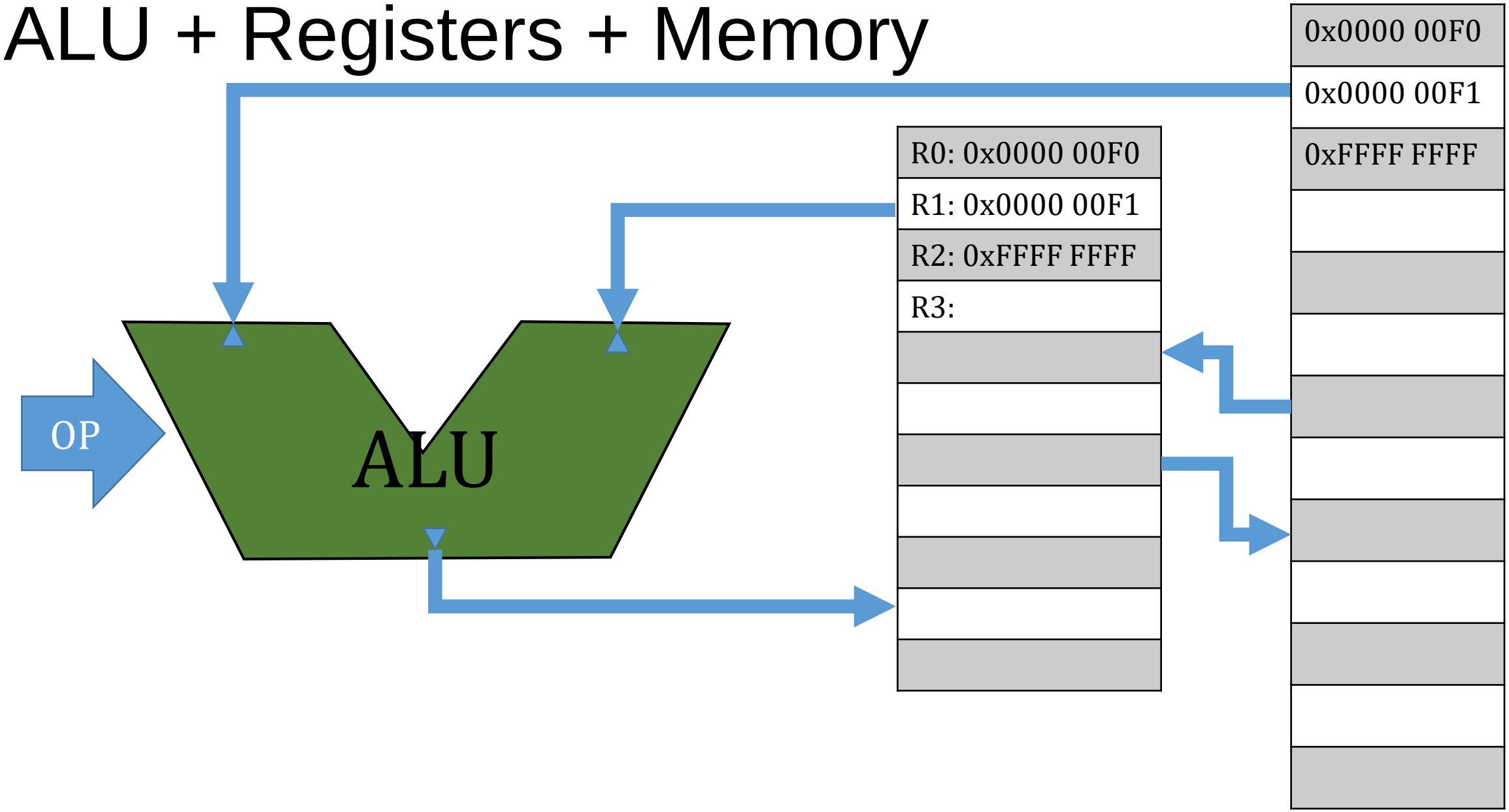
Arithmetic Logic Unit (ALU)



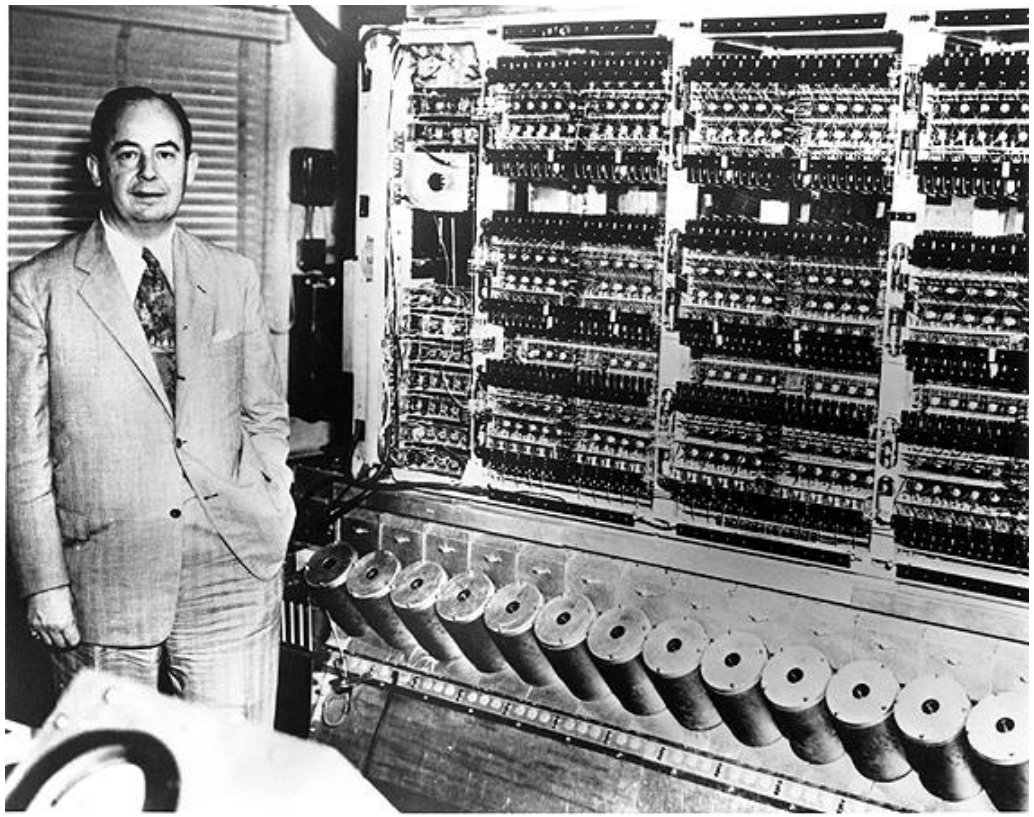
ALU + Registers



ALU + Registers + Memory

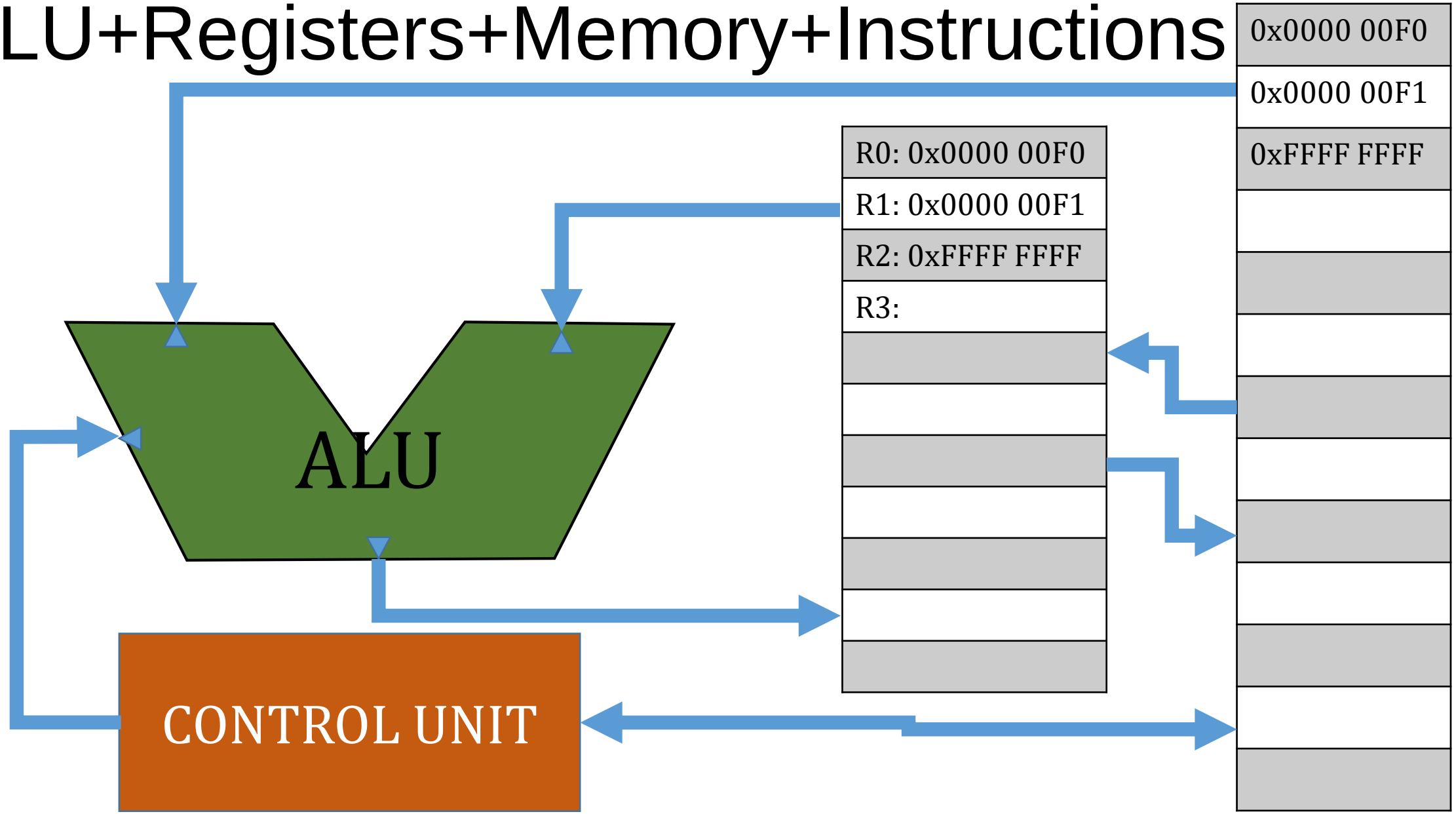


Computer vs. Adding Machine



What is the difference?

ALU+Registers+Memory+Instructions



How do you describe your machine to a software guy?

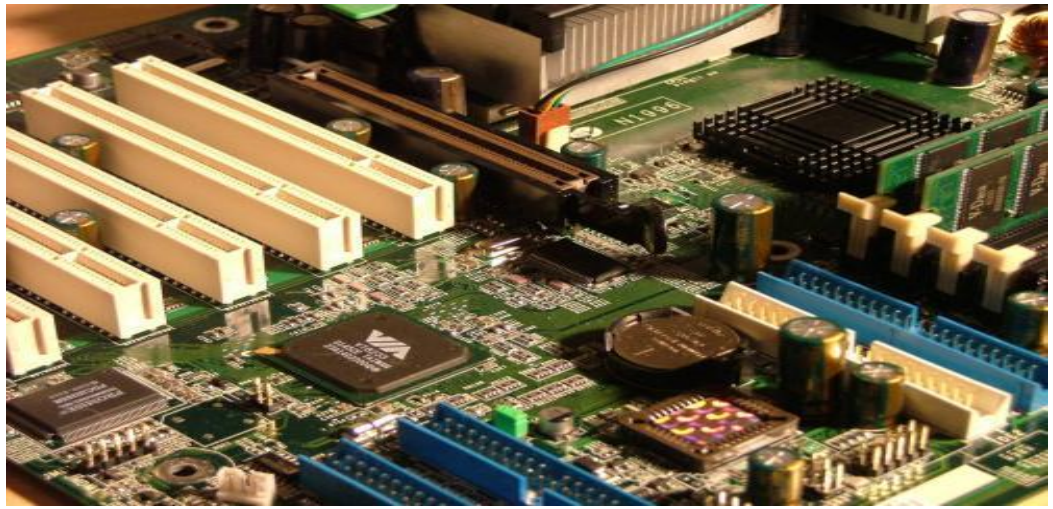
"Instruction Set Architecture" (ISA)

```
SUB32   PROC          ; procedure begins here
        CMP   AX,97   ; compare AX to 97
        JL    DONE    ; if less, jump to DONE
        CMP   AX,122  ; compare AX to 122
        JG    DONE    ; if greater, jump to DONE
        SUB   AX,32   ; subtract 32 from AX
DONE:   RET           ; return to main program
SUB32   ENDP          ; procedure ends here
```

I S A

ISA Contents

- The instructions the hardware recognizes
  - add, move, get, …
- The data types the instructions can work on
  - two's complement binary, ascii character, unsigned binary, etc.
- The data the instructions can work on
  - Registers
  - Memory
- The external interfaces supported by the instructions
  - File I/O
  - Exception Handling and Interrupts



ISA Compatability

- It's nice to have the same software run on different hardware
- It's nice to have the apps I wrote for iPhone5 run on iPhone6
- But different hardware has different capabilities
- I can do more things on an iPhone6 than an iPhone5

- How can we manage this change?

Downward Compatability

- If something is downward compatible
  - It will do everything it used to do
  - It may be able to do more

- Implications …
  - Things can only get more complex
  - Every “mistake” made early propagates forever
  - Eventually, things get really complicated

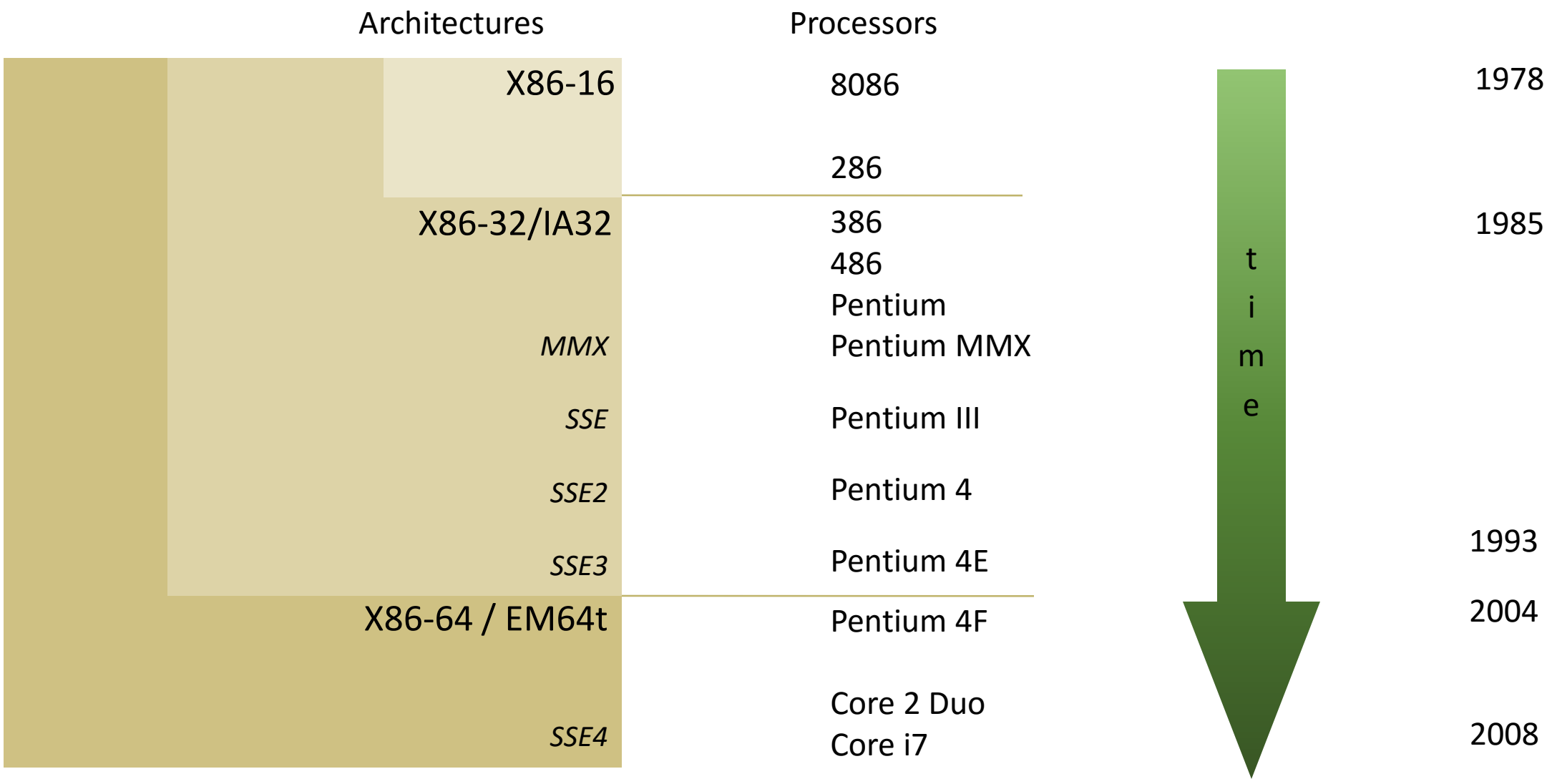
Example Downward Compatability

- 1972 – Intel 8008 – 8 bit word – terminals & calculators
  - Registers: 16 bits with a “low” byte and a “high” byte al,ah; bl, bh; …
- 1978 – Intel 8086 – 16 bit word – Vanilla PC
  - Registers: 16 bits: ax (al,ah); bx (bl,bh) …
- 1985 – Intel 80386 – 32 bit word - PC/XT - Windows
  - Registers: 32 bits: eax [ax (al,ah)]; ebx [bx (bl, bh)]; …
- ~2000 – AMD AMD64 – 64 bit word
  - Registers: 64 bits: rax { eax [ax (al,ah)]}; rbx {ebx [bx (bl,bh)] }; …

Intel Microprocessor Milestones

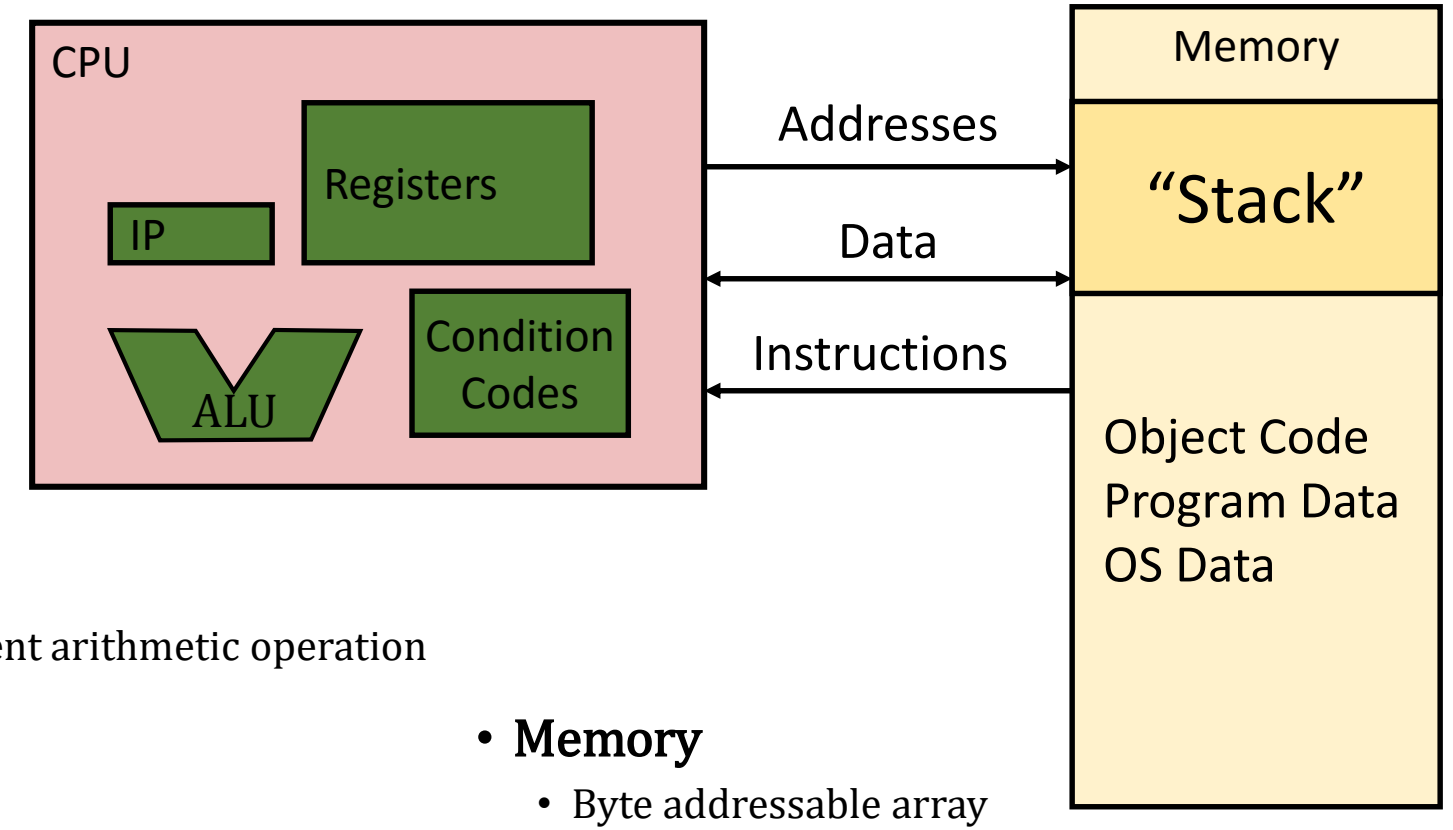
Date	Chip	Data Width	Transistors	Clock Rate	Usage
1972	8008	8	3.5K	0.8 MHz	Monitors, Controllers
1974	8080	8 (some 16)	6K	2 MHz	+ Calculators
1978	8086	16	29K	10 MHz	IBM PC/DOS – First x86
1985	80386	32	275K	40 MHz	Windows+Linux
1993	Pentium P5	32	3.1M	66 MHz	ditto
2004	Pentium 4F	64	125M	3.8 GHz	ditto
2008	Core i7	64	731M	3.33 GHz	ditto
2014	Core i7 extreme	64	1.4B	3.5 GHz	ditto

37 Years of Intel x86 Evolution



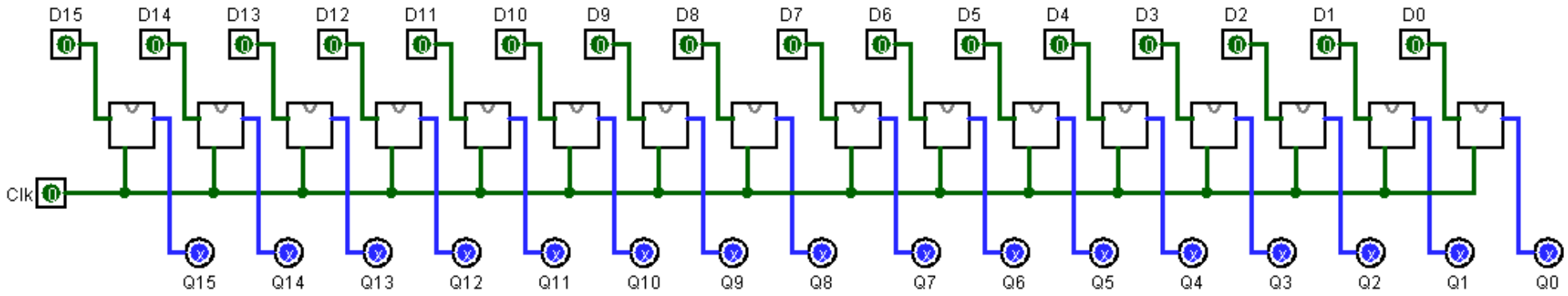
Assembly Programmer's View of x86

- Programmer-Visible State
  - IP: Instruction Pointer
    - Address of next instruction
    - Called "EIP" (IA32) or "RIP" (x86-64)
  - Register file
    - Heavily used program data
  - Condition codes
    - Store status information about most recent arithmetic operation
    - Used for conditional branching
- Arithmetic/Logic Unit (ALU)
  - Performs Instructions



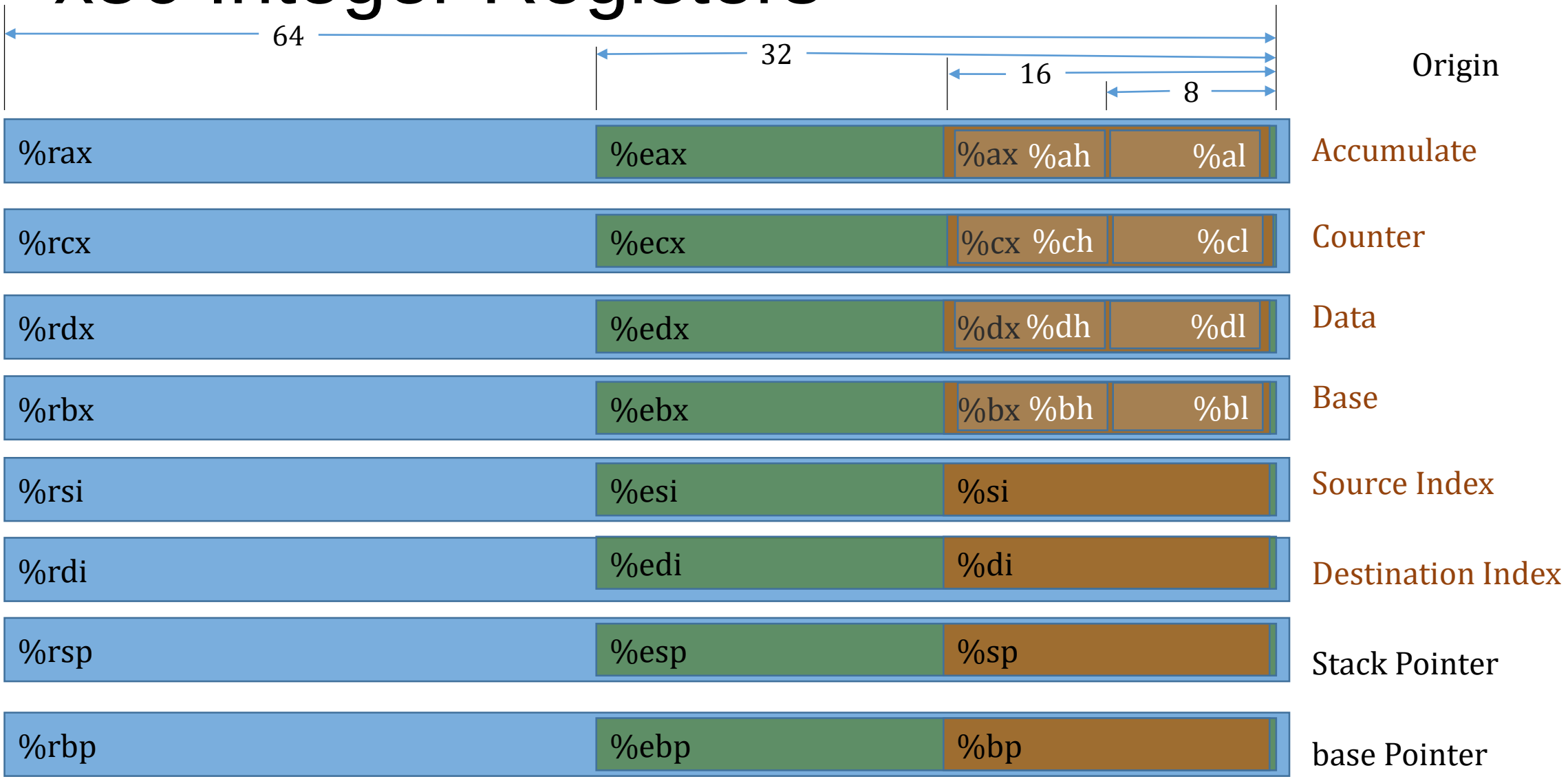
- **Memory**
  - Byte addressable array
  - Code, user data, (some) OS data
  - Includes stack used to support procedures

X86 Integer Registers



- 8+ 64 bit Registers
- Fast Read/Write (CPU Speed)
- No explicit data type!
- Values undefined (X) until set
- Register Names based on Historic Usage
  - Naming conventions used to access portions of registers

x86 Integer Registers



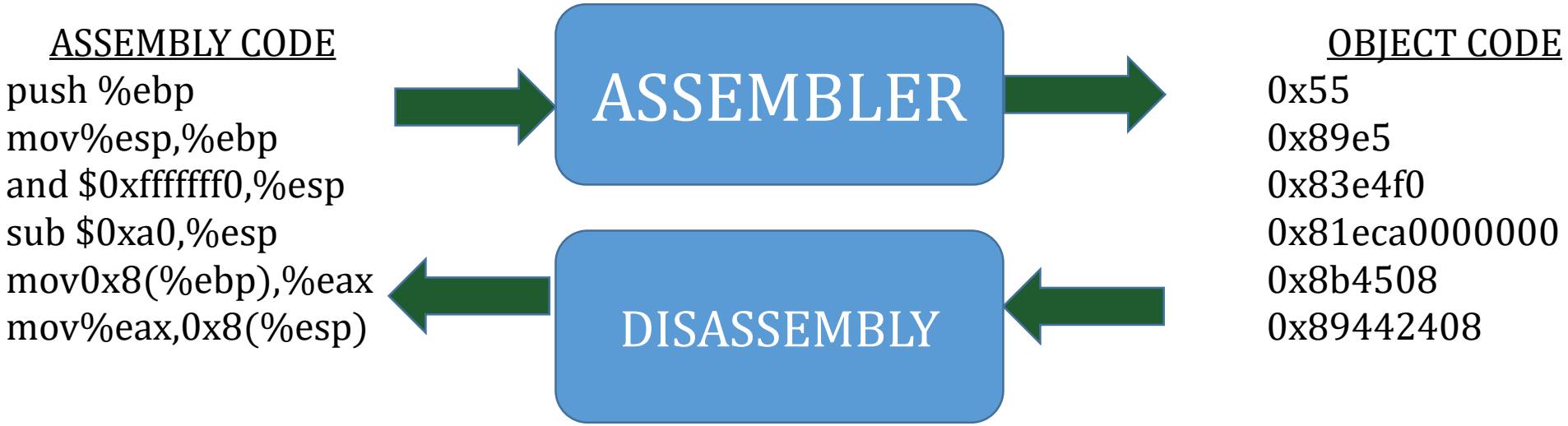
What is "Memory"

- In x86, memory is a list of bytes
- Each byte of memory has a specific ADDRESS… the index of the byte from the beginning of memory
- For this class, we will use 32 bit addresses
- Memory read/write takes about 10x longer than register read/write!
- Memory read/write is about 10x faster than read/write from disk (external IO)

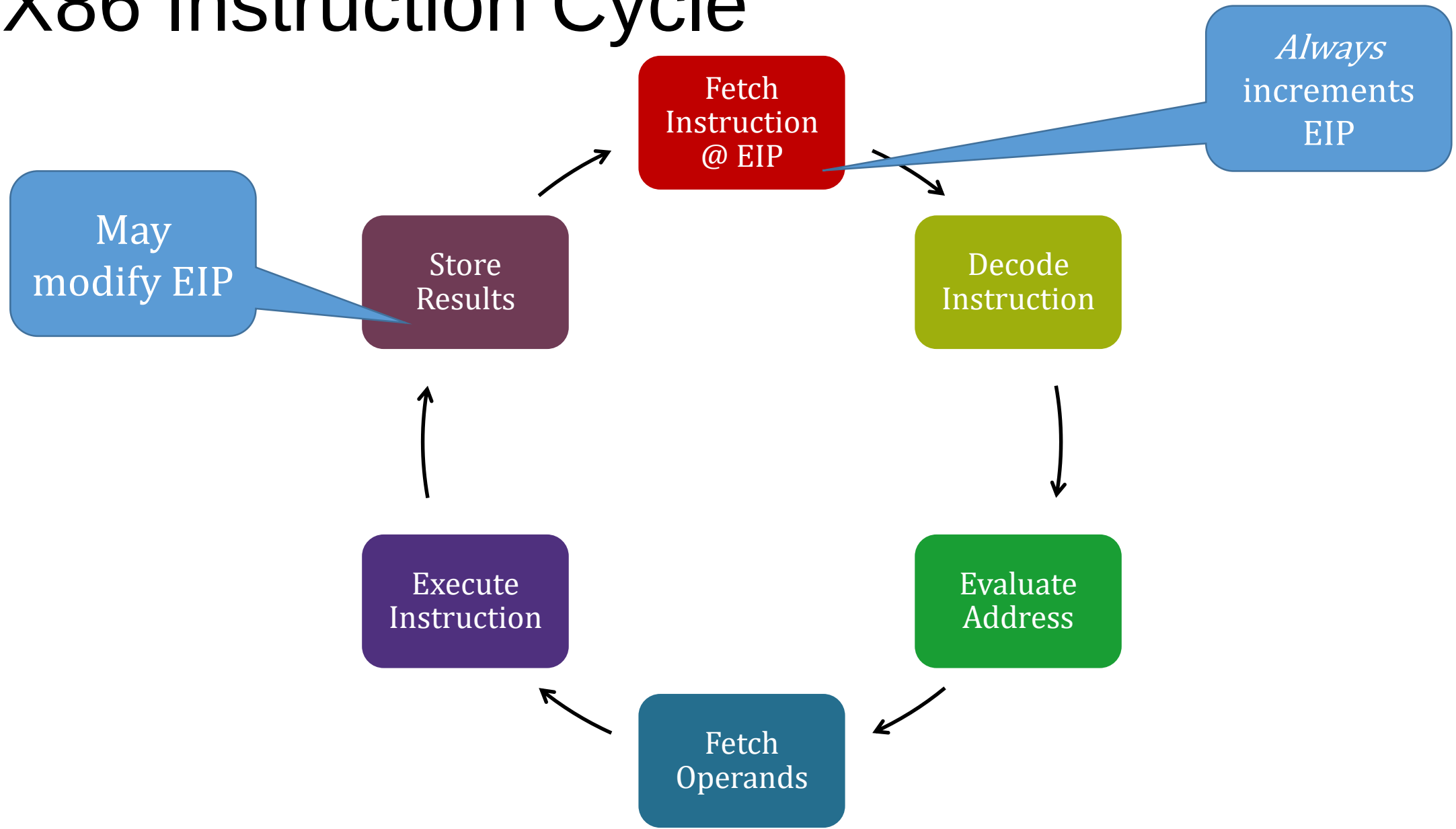
Address	Value
0xFFFF FFFF	
0xFFFF FFFE	0xDA
0xFFFF FFFD	0xED
0xFFFF FFFC	0xBE
0xFFFF FFFB	0xEF
…	
0x0000 0C08	0x00
0x0000 0C07	0x01
0x0000 0C06	0x18
0x0000 0C05	0x00
….	
0x0000 0003	0x00
0x0000 0002	0x00
0x0000 0001	0x00
0x0000 0000	0x03

X86 Instructions

- Smallest (Atomic) directive to x86 “hardware”
- Consist of Opcode and Operands
- Two Flavors
  - Man-readable “Assembly”
  - Machine Readable “Object Code” or “Machine Code” or “Binary”
- Translation…



X86 Instruction Cycle



X86 Instruction Cycle