Announcements

- Assignment 1
  - To be returned this afternoon
- Progress Meeting
  - October 6, 8, and 10
  - A 15-minute meeting to discuss your progress and learning goals
- Assignment 2
  - Posted today, due October 10
  - Simulation: Building an accumulator-based system

Lecture 12

Representation of Programs

CPSC 275
Introduction to Computer Systems

Intel x86 Processors

- Dominate laptop/desktop/server market
- Evolutionary design
  - Backwards compatible up until 8086, introduced in 1978
  - Added more features as time goes on
- Complex instruction set computer (CISC)
  - Many different instructions with many different formats
  - Hard to match performance of *Reduced Instruction Set Computers* (RISC)
    - Mainly used for supercomputers. Now for mobile systems

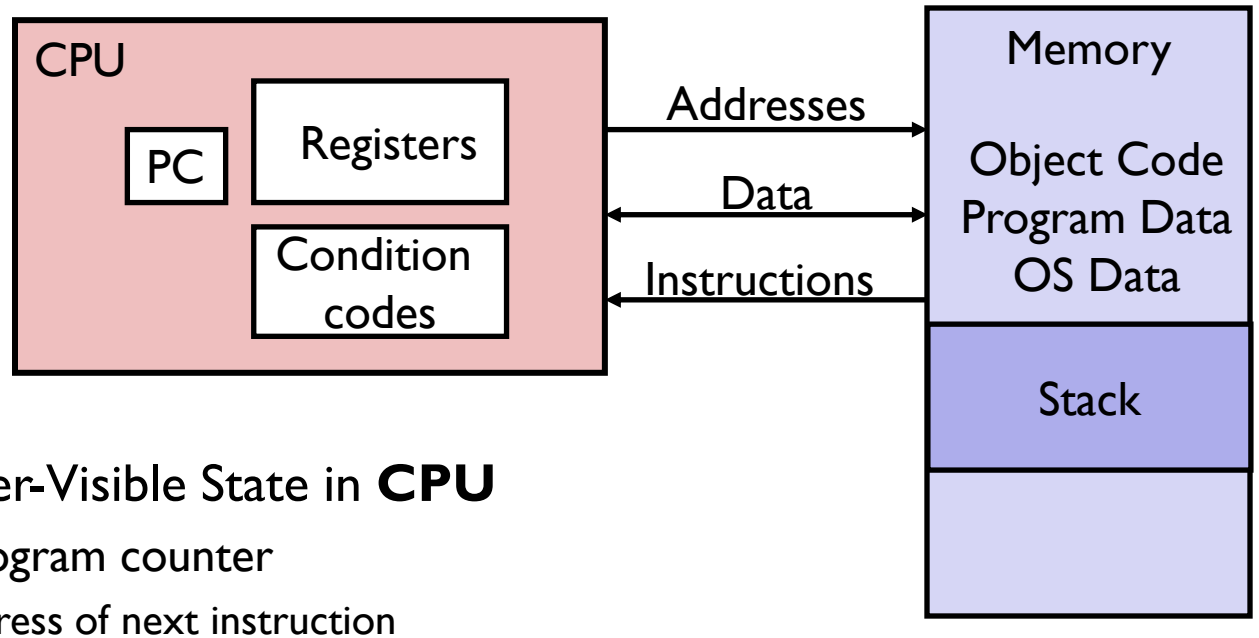
Intel x86 Evolution: Milestones

Name	*Date*	*Transistors*	*MHz*
▪ 8086	1978	29K	5-10
• First 16-bit Intel processor. Basis for IBM PC & DOS			
• 1MB address space			
▪ 386	1985	275K	16-33
• First 32 bit Intel processor, referred to as *IA32*			
• Added *flat addressing*, capable of running Unix			
▪ Pentium 4E	2004	125M	2800-3800
• First 64-bit Intel x86 processor, referred to as *x86-64*			
▪ Core 2	2006	291M	1060-3500
• First multi-core Intel processor			
▪ Core i7	2008	731M	1700-3900
• Four cores			
…			
▪ Core i9	2023	25B	3200-5800
• 24 cores			

Some Definitions

- Instruction set architecture (ISA)
  - Parts of a processor design that one needs to understand to write assembly code.
  - Examples: instruction set specification, registers.
- Microarchitecture: Implementation of the ISA
  - Examples: cache sizes and core frequency.
- Example ISAs (Intel): x86, **IA-32**

Programmer's View on Computer System



Programmer-Visible State in **CPU**

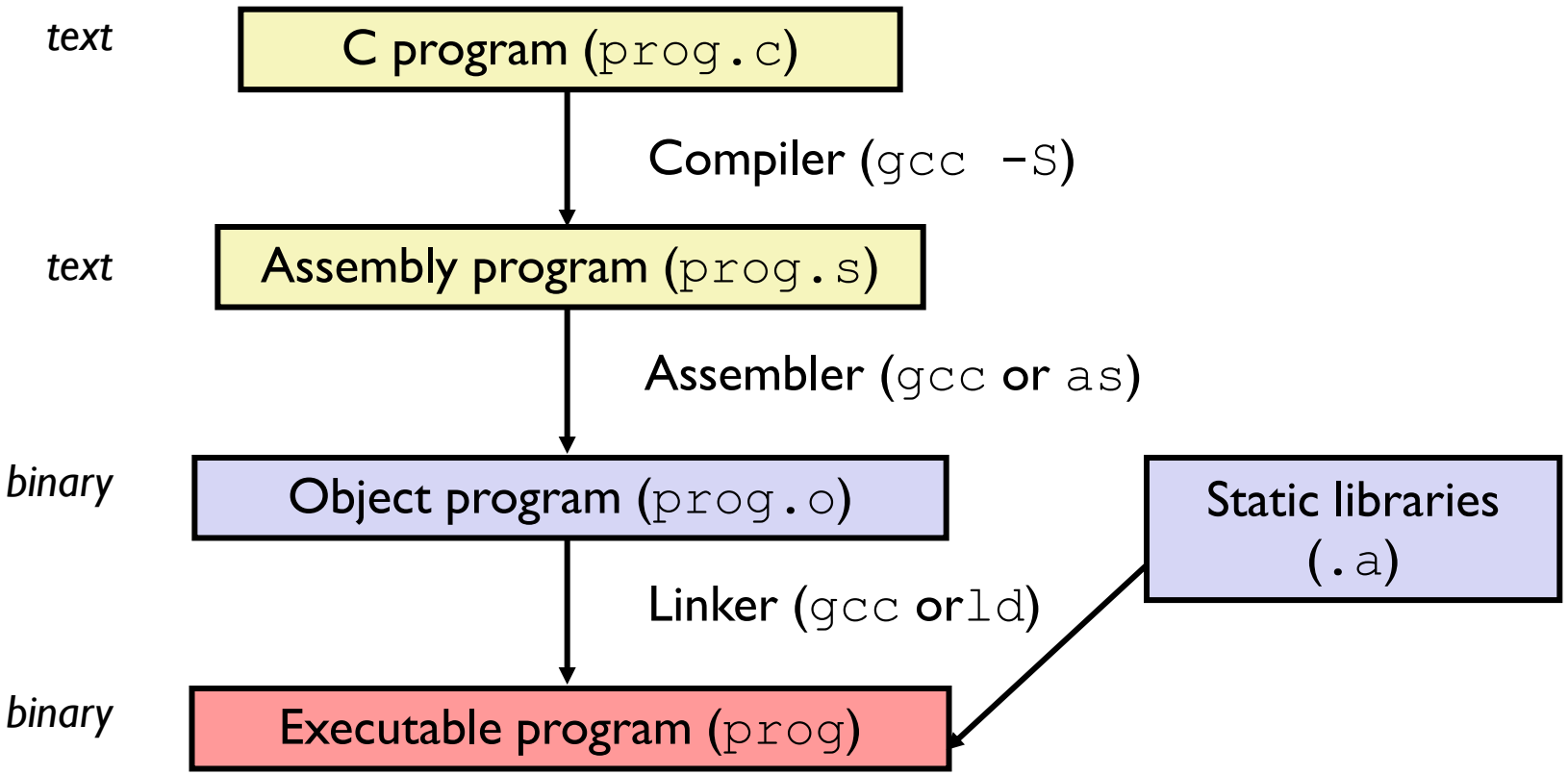
- PC: Program counter
  - Address of next instruction
  - Called `EIP` (IA32)
- Register file
  - Heavily used to store program/data
- Condition codes
  - Store status information about most recent arithmetic operation
  - Used for conditional branching

Memory

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

Turning C into Object Code

- Source **`prog.c`**
- Compile with command: **`gcc –m32 -o prog prog.c`**



Compiling Into Assembly

C Code

```
int sum(int x, int y)
{
  int t = x + y;
  return t;
}
```

Generated IA-32 Assembly

```
sum:
    pushl   %ebp
    movl    %esp,%ebp
    movl    12(%ebp),%eax
    addl    8(%ebp),%eax
    popl    %ebp
    ret
```

Obtain with command

```
gcc -m32 -O1 -S code.c
```

Produces file `code.s` where `-O1` indicates level-one optimization.