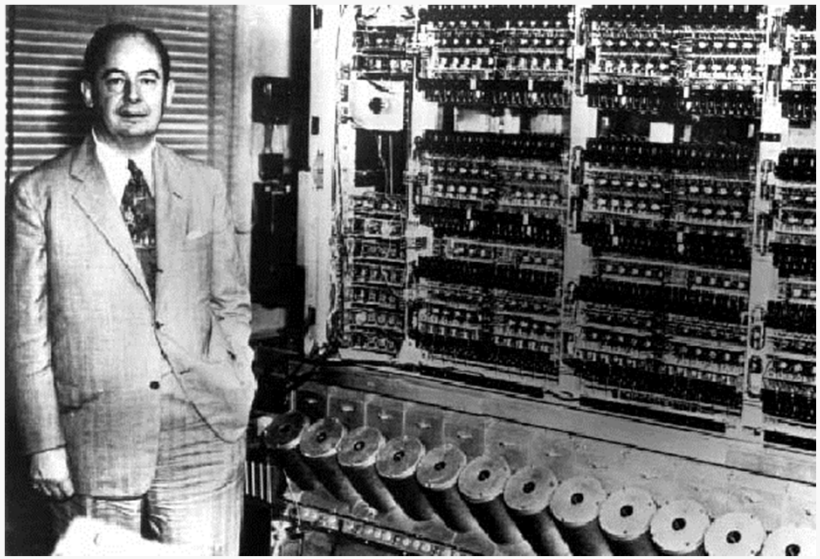
The von Neumann Machine

1945: John von Neumann

- Wrote a report on the stored program concept, known as the First Draft of a Report on EDVAC
- also Alan Turing… Konrad Zuse… Eckert & Mauchly…

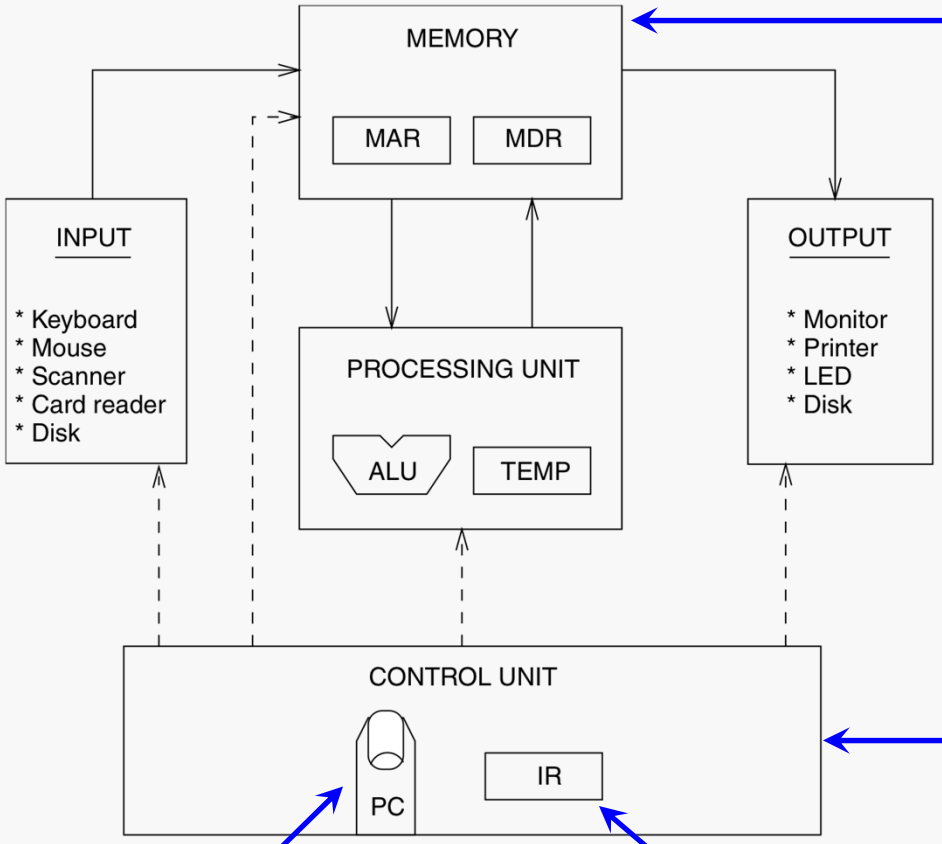
The basic structure proposed in the draft became known as the "von Neumann machine" (or model).

- a memory, containing instructions and data
- a processing unit, for performing arithmetic and logical operations
- a control unit, for interpreting instructions



The von Neumann Machine

Abstraction of von Neumann Architecture



stores both program instructions and data

decodes current instruction,

manages processing unit to carry out instruction

***program counter*: points to the next instruction to be fetched**

***instruction register*: stores current instruction**

Intel x86 History

Totally 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
  - But, only small subset encountered with Linux programs
- Hard to match performance of Reduced Instruction Set Computers (RISC)
- But, Intel has done just that!
  - In terms of speed. Less so for low power.

Intel x86 History

Name	*Date*	*Transistors*	*MHz*
8086	1978	29K	5-10

- First 16-bit processor. Basis for IBM PC & DOS
- 1MB address space

80386	1985	275K	16-33

- First 32 bit processor , referred to as IA32
- Added "flat addressing"
- Capable of running Unix
- 32-bit Linux/gcc uses no instructions introduced in later models

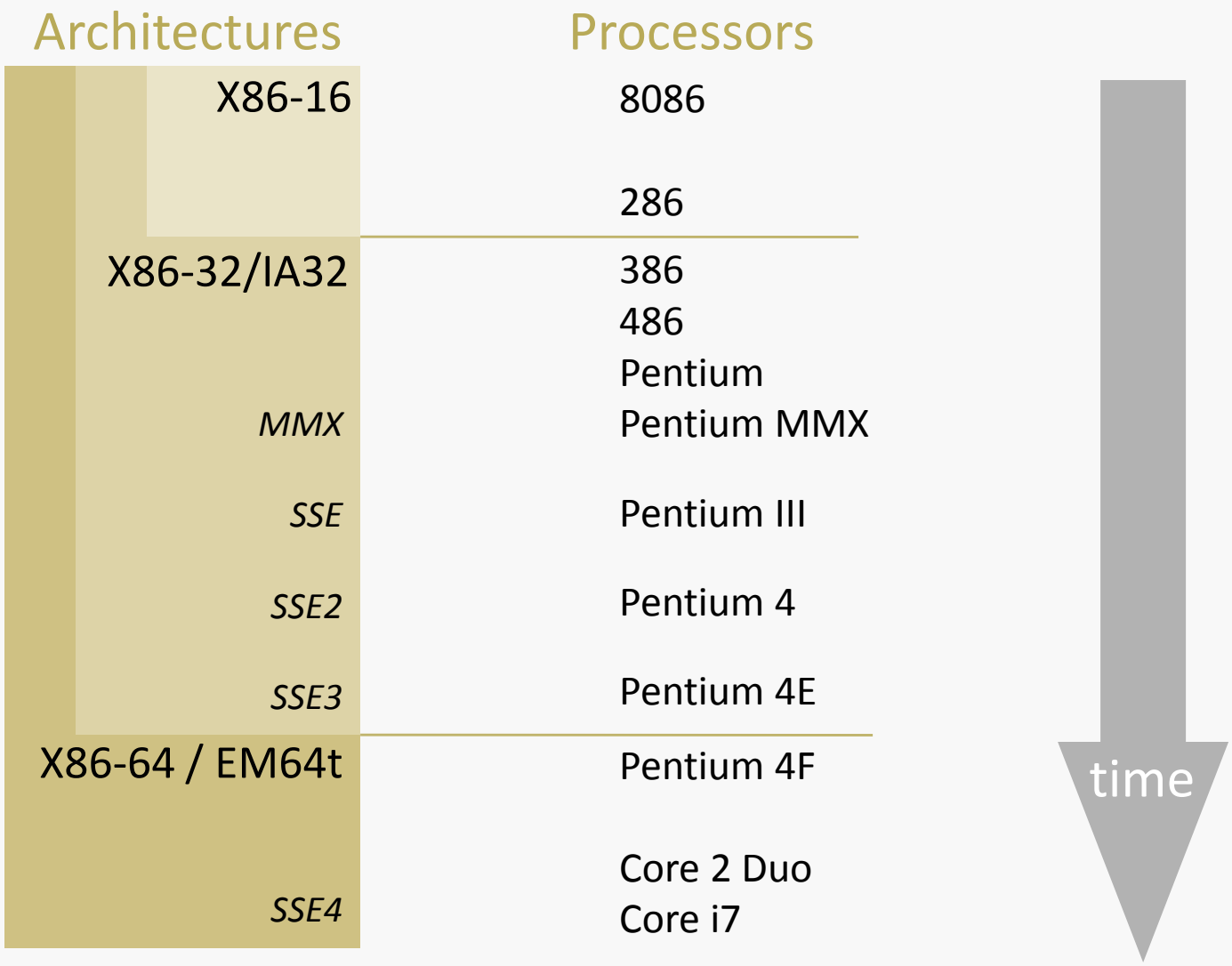
Pentium 4F	2004	125M	2800-3800

- First 64-bit processor, referred to as x86-64

Core i7	2008	731M	2667-3333

- Our shark machines

Intel x86 History



Intel x86 History

Intel Attempted Radical Shift from IA32 to IA64

- Totally different architecture (Itanium)
- Executes IA32 code only as legacy
- Performance disappointing

AMD Stepped in with Evolutionary Solution

- x86-64 (now called "AMD64")

Intel Felt Obligated to Focus on IA64

- Hard to admit mistake or that AMD's approach is better

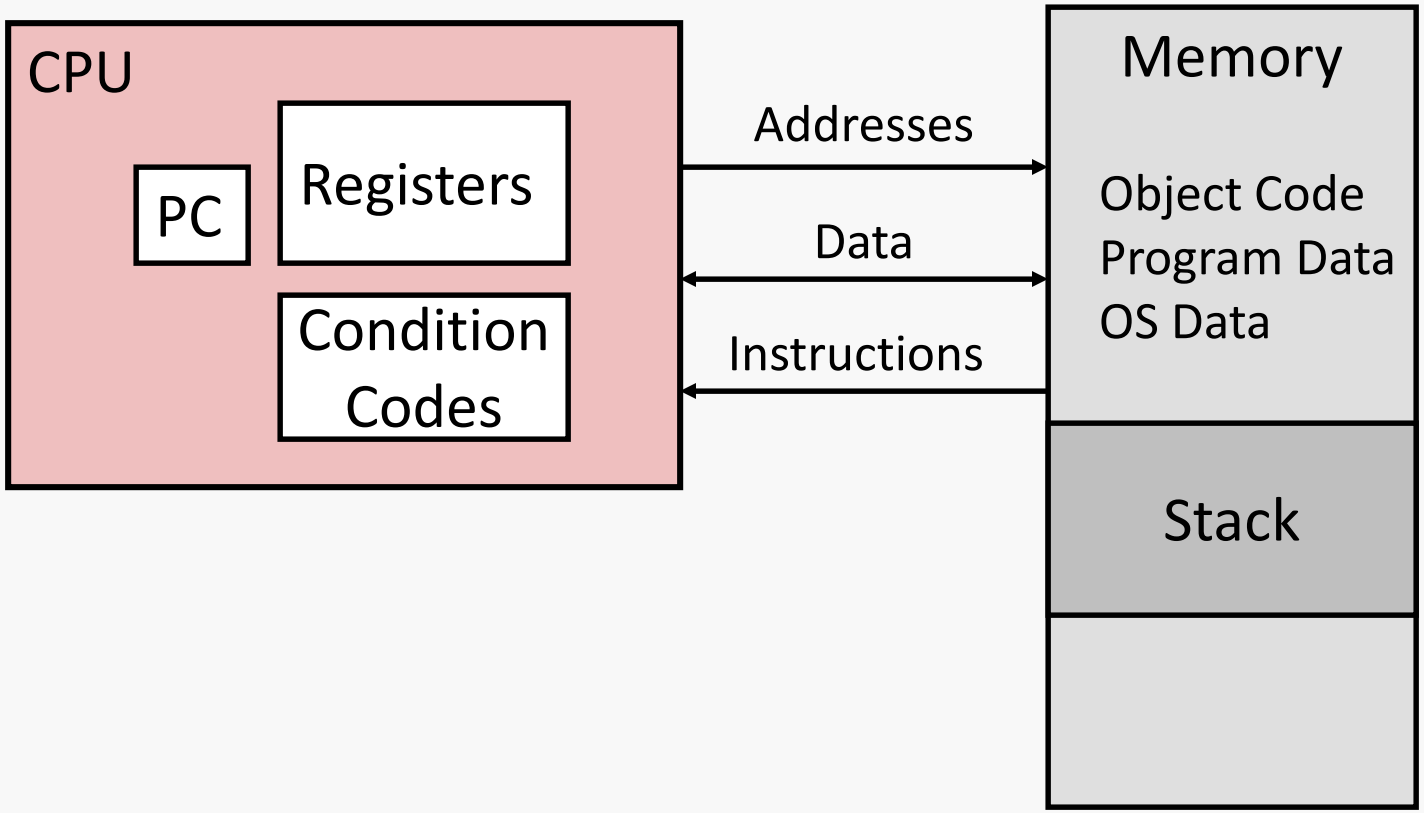
2004: Intel Announces EM64T extension to IA32

- Extended Memory 64-bit Technology
- Almost identical to x86-64!

All but low-end x86 processors support x86-64

- But, lots of code still runs in 32-bit mode

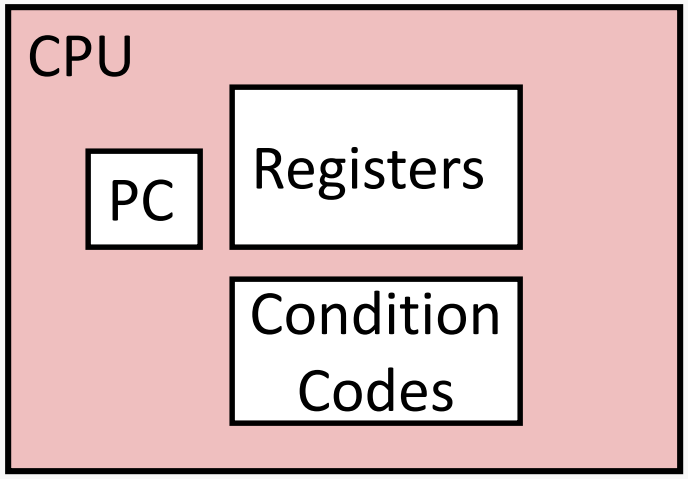
von Neumann View of x86



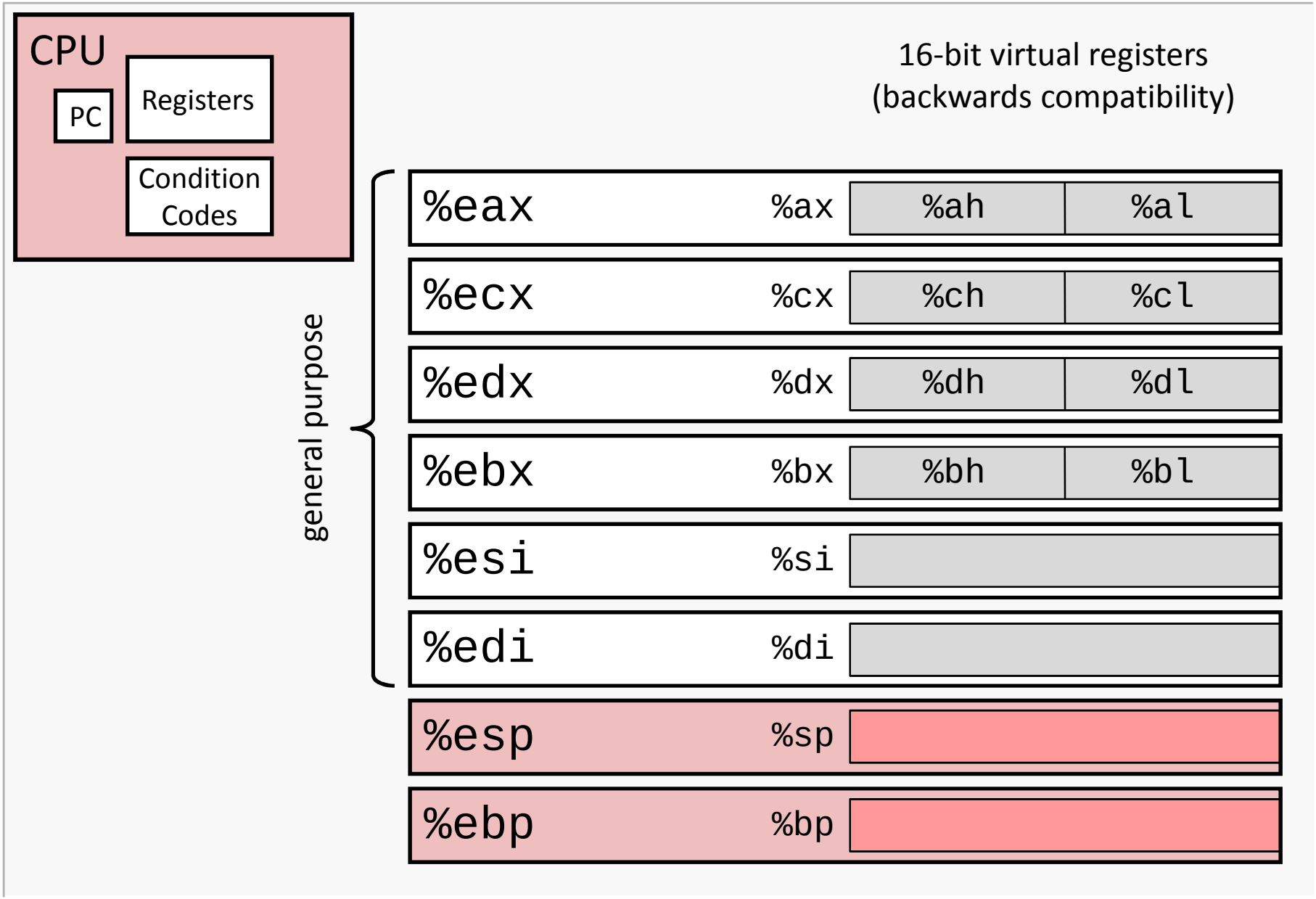
High-level x86 CPU

Programmer-Visible State

- PC: Program counter
  - 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

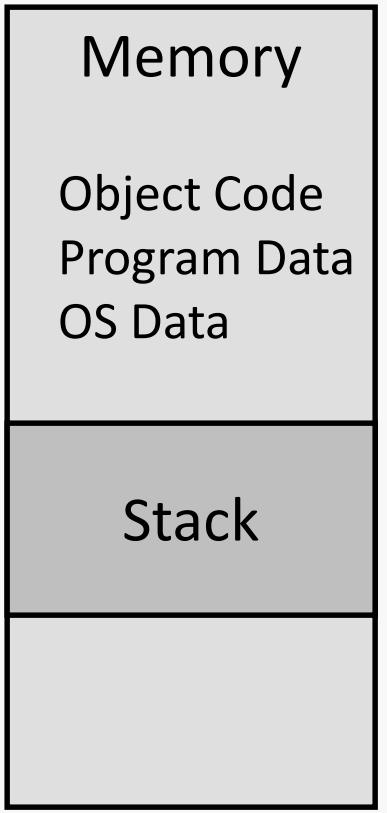


IA32 Integer Registers



High-level x86 Memory

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

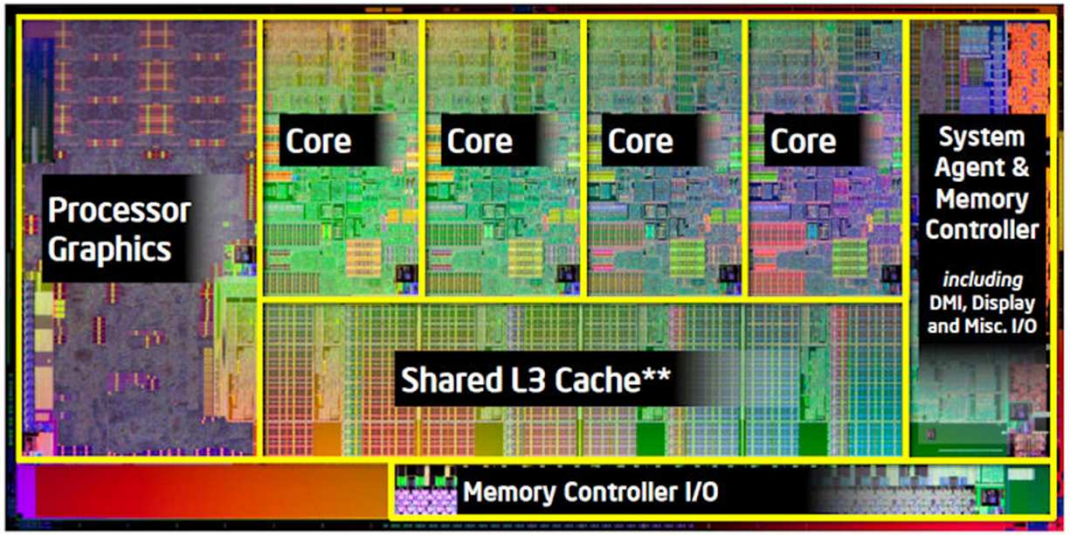


Programming the Machine

```
// C code
. . .
int imax(int first, int second) {

   if ( first >= second )
      return first;
   return second;
}
```

But the hardware only "understands" binary representations



Programming the Machine

```
int imax(int first, int second) {

   if ( first >= second )
      return first;
   return second;
}
```

```
gcc  -O0 -c -m32 -std=c99imax.c
```

```
457f464c0101000100000000000000000
0001000300010000000000000000000000
00bc0000000000003400000000000028
000900068955 8be50845453b7c0c8b05
084503eb458b5d0c00c3000047004343
203a55287562746e2f75694c616e6f72
3420352e322e382d62756e7575742934
. . .
```

But who wants to program in binary?

Programming the Machine

```
int imax(int first, int second) {

   if ( first >= second )
      return first;
   return second;
}
```

```
. . .
imax:
        pushl   %ebp
        movl    %esp, %ebp

        movl    8(%ebp), %eax
        cmpl    12(%ebp), %eax
        jl      .L2
        movl    8(%ebp), %eax
        jmp     .L3
.L2:
        movl    12(%ebp), %eax
.L3:
        popl    %ebp
        ret
. . .
```

Solution:

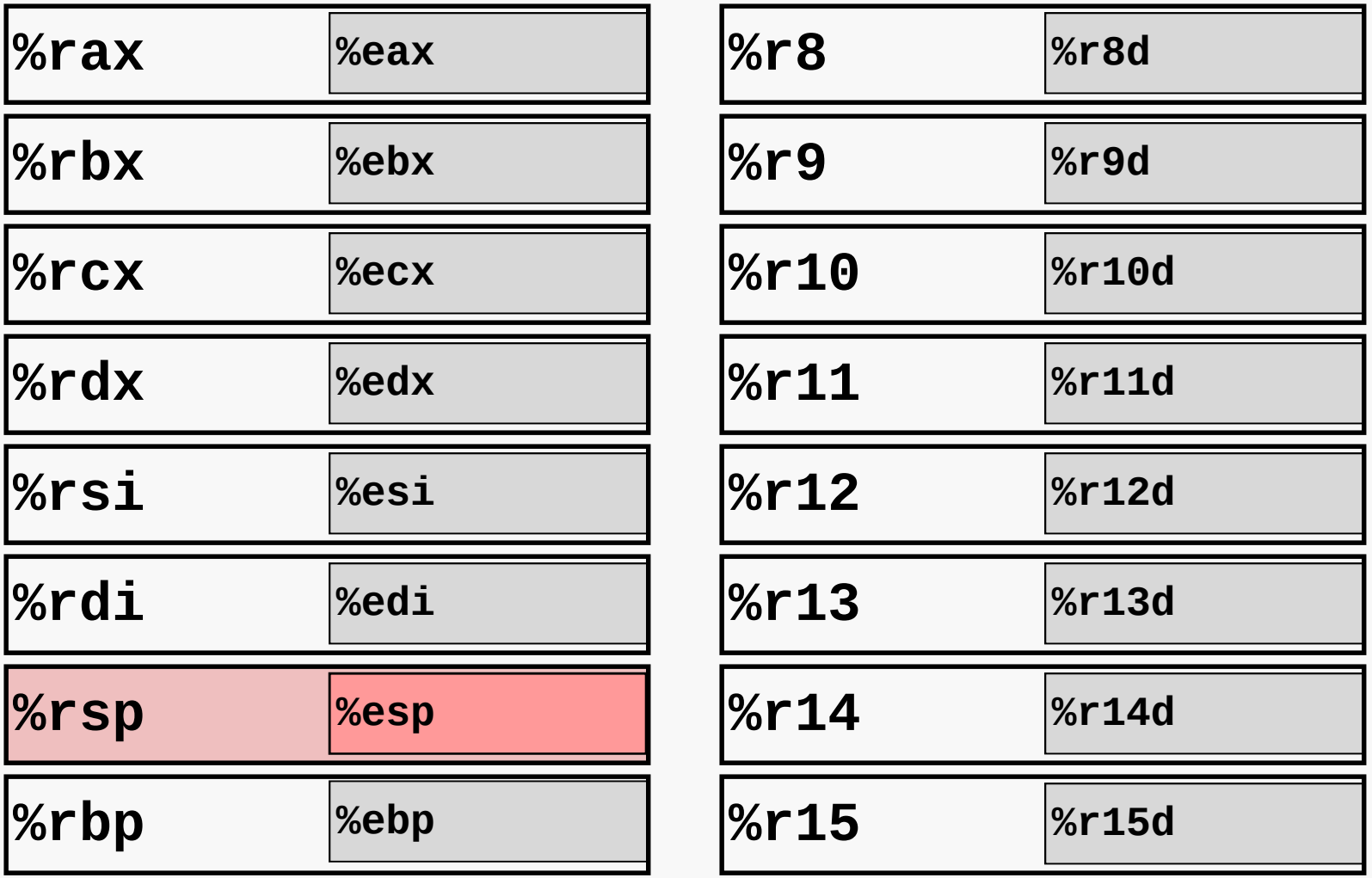
translate high-level language code into intermediate-level code which is more human-friendly,

then translate that "assembly" code into the machine's langauge.

Explanation: moving data

	Source	Dest	Src,Dest	C Analog
movl	*Imm*	*Reg*	`movl $0x4,%eax`	`tmp = 0x4;`
		Mem	`movl $-147,(%eax)`	`*p = -147;`
	Reg	*Reg*	`movl %eax,%edx`	`tmp2 = tmp1;`
		Mem	`movl %eax,(%edx)`	`*p = tmp;`
	Mem	*Reg*	`movl (%eax),%edx`	`tmp = *p;`

x86-64 Integer Registers



- Extend existing registers. Add 8 new ones.
- Make **%ebp/%rbp** general purpose

x86-64 Example

```
int imax(int first, int second) {

   if ( first >= second )
      return first;
   return second;
}
```

```
. . .
    movl    %edi, -4(%rbp)
    movl    %esi, -8(%rbp)
    movl    -4(%rbp), %eax
    cmpl    -8(%rbp), %eax
    jl      .L2
    movl    -4(%rbp), %eax
    jmp     .L3
.L2:
    movl    -8(%rbp), %eax
.L3:
. . .
```