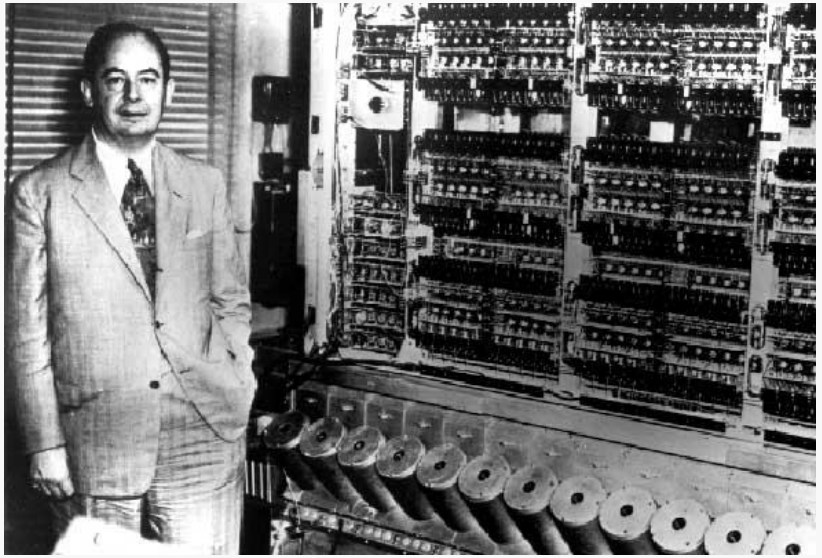
The von Neumann Machine

1945: John von Neumann

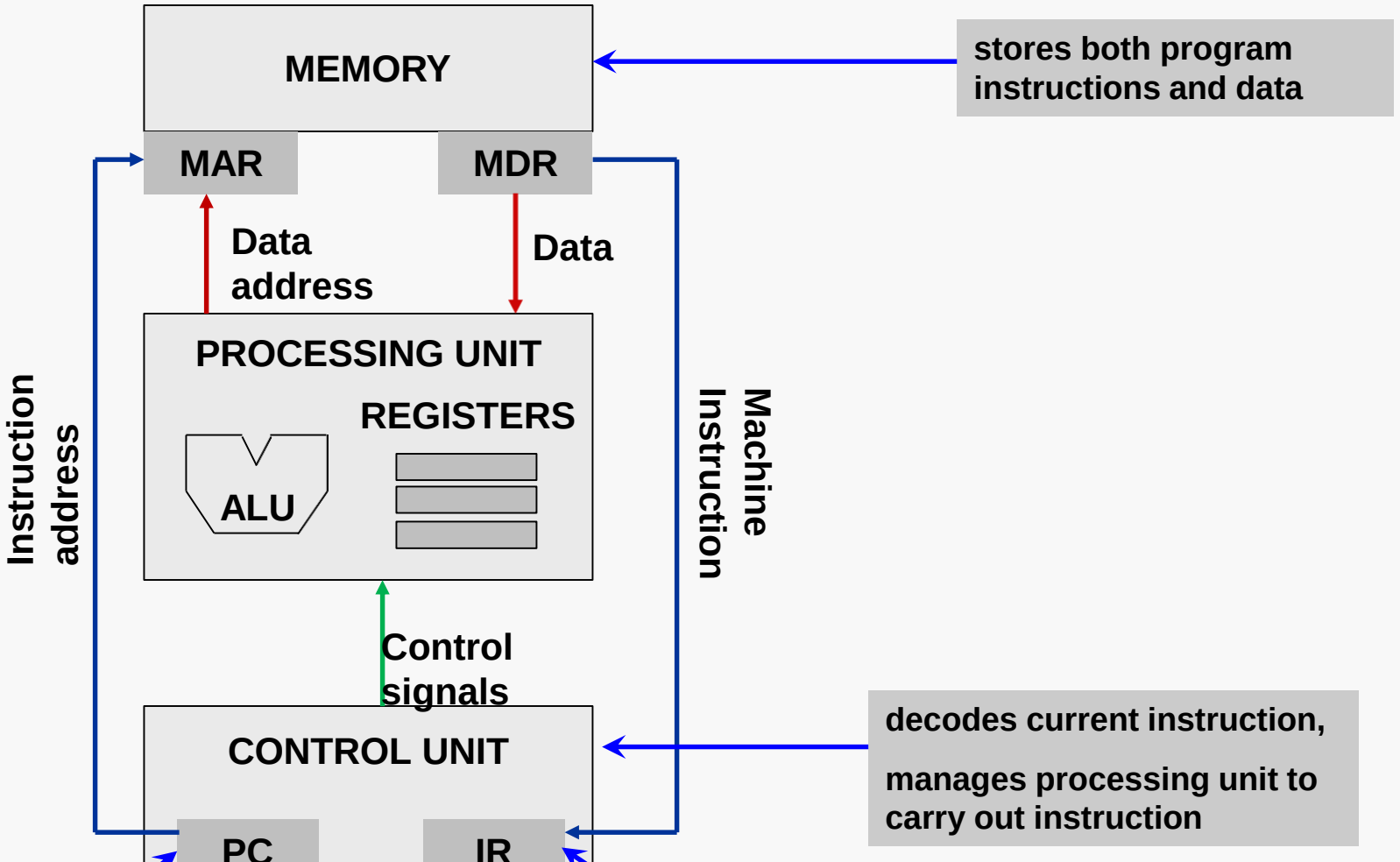
- Wrote a report on the stored program concept, known as the *First Draft of a Report on EDVAC*
- also Alan Turing… Konrad Zuse… Eckert & Mauchly…

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



Intel x86 Family

Totally dominate laptop/desktop/server market

Evolutionary design

- Backwards compatible to the 8086, introduced in 1978
- Open architecture: 3rd party suppliers for all sorts of external hardware and software
- Added more features as time went on

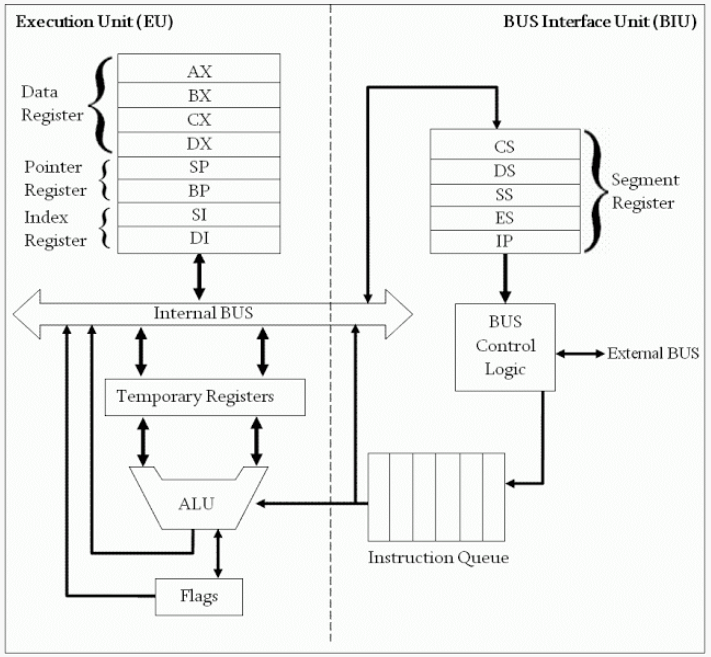
Complex instruction set computer architecture (CISC)

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

Intel x86 History

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

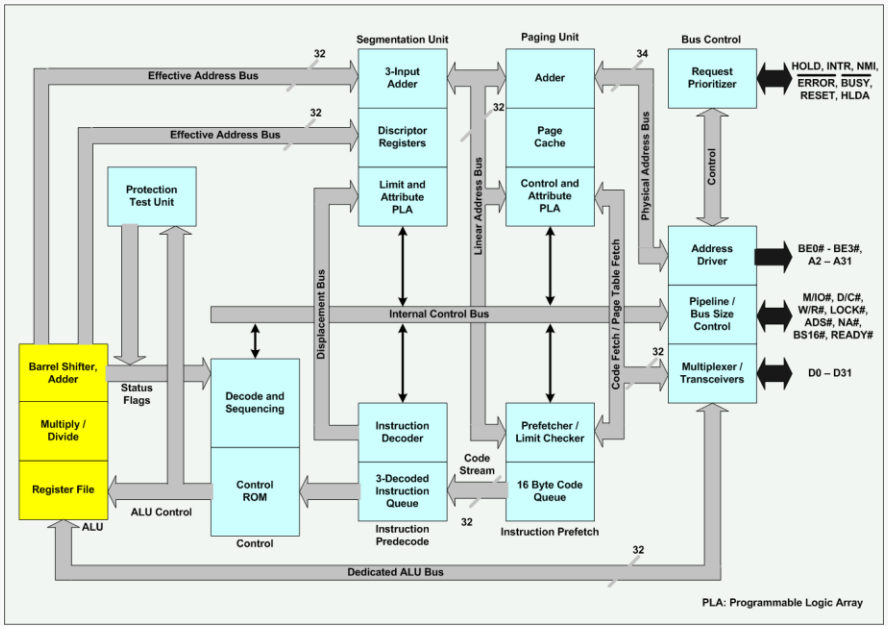
- First 16-bit processor. Basis for IBM PC & DOS
- 1MB address space
- 8088: slight modification of 8086 chip used in IBM PC
- Quickly dominated personal computer market in sales via IBM PC



Intel x86 History

Name	*Date*	*Transistors*	*Clock rate*
80386	1985	275K	16-33 MHz

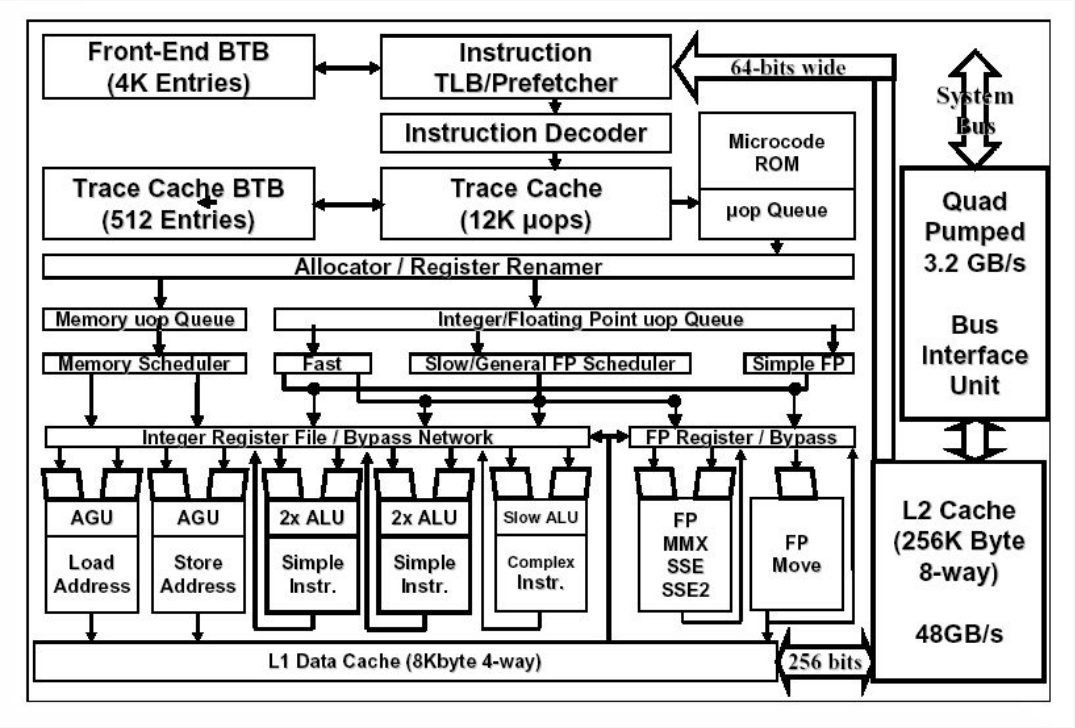
- First 32 bit processor , referred to as IA32 architecture
- Added “flat addressing”
- Capable of running Unix
- 32-bit Linux/gcc used no instructions introduced in later models
- IBM PC superceded by 386-based machines from other vendors; explosion of "clone" vendors



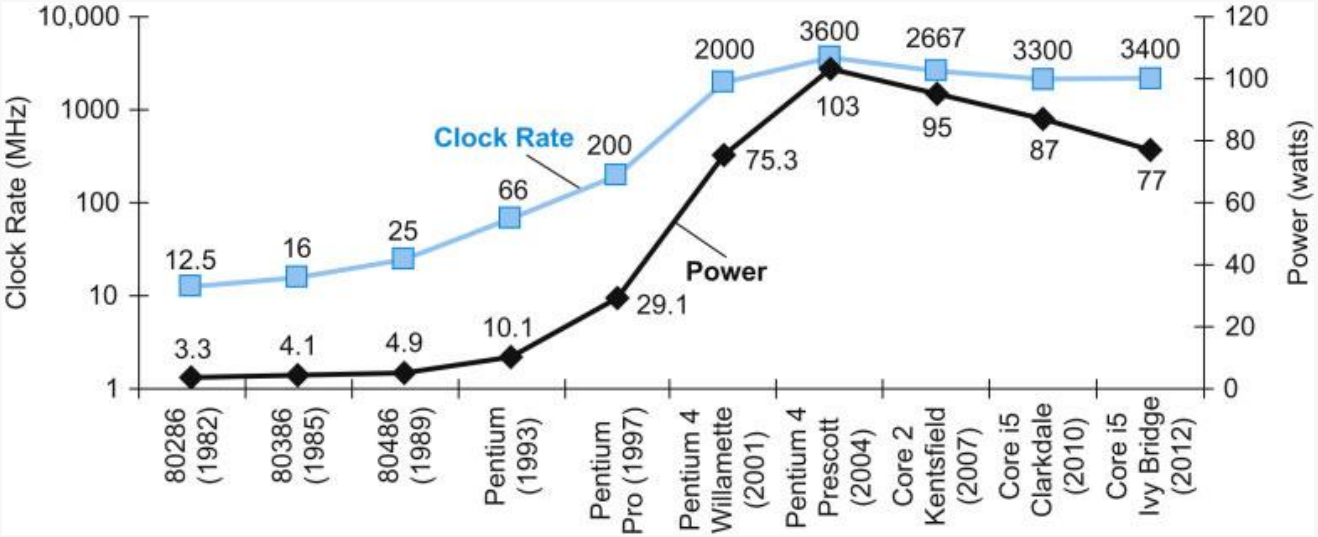
Intel x86 History

Name	*Date*	*Transistors*	*Clock rate*
Pentium 4F	2004	125M	2800-3800 MHz

- First (successful) 64-bit processor (from Intel), referred to as x86-64
- Based on AMD 64-bit architecture, which was derived from IA-32



The Power Wall



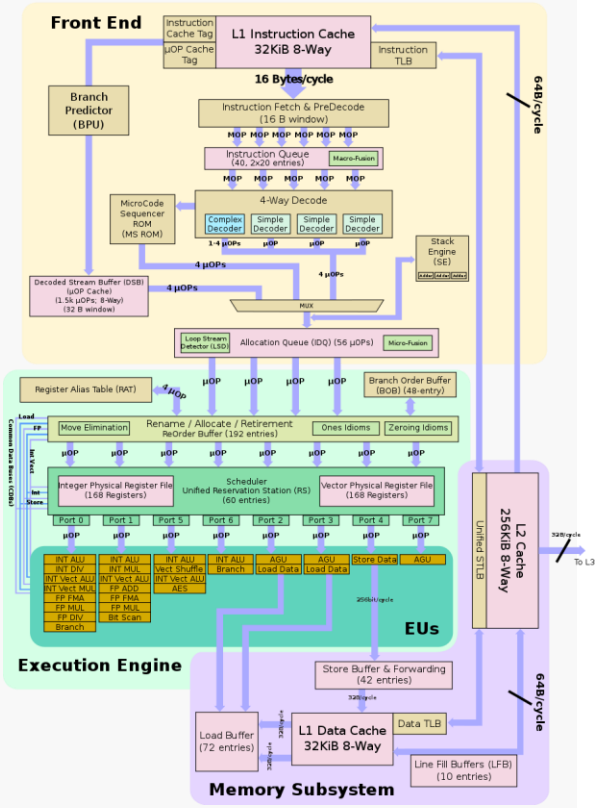
"Shortly after the beginning of the 21st century an inflection point was reached: in a speech in 2001, and Intel executive pointed out that an extrapolation of increasing power density (and thus temperature) of microprocessor chips would exceed that of a rocket nozzle by 2006, and of the sun's surface by 2012."

"Clearly something had to give, and the result is that going forward we can expect an exponential increase in the number of processors (cores) available to us, but not in their individual performance."

Intel x86 History

Name	*Date*	*Transistors*	*Clock rate*
Core i7	2008	731M	2667-3333 MHz

- Multi-core architecture (4, 6, 8 cores)
- Reduced clock rates compensated for by multiple cores



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

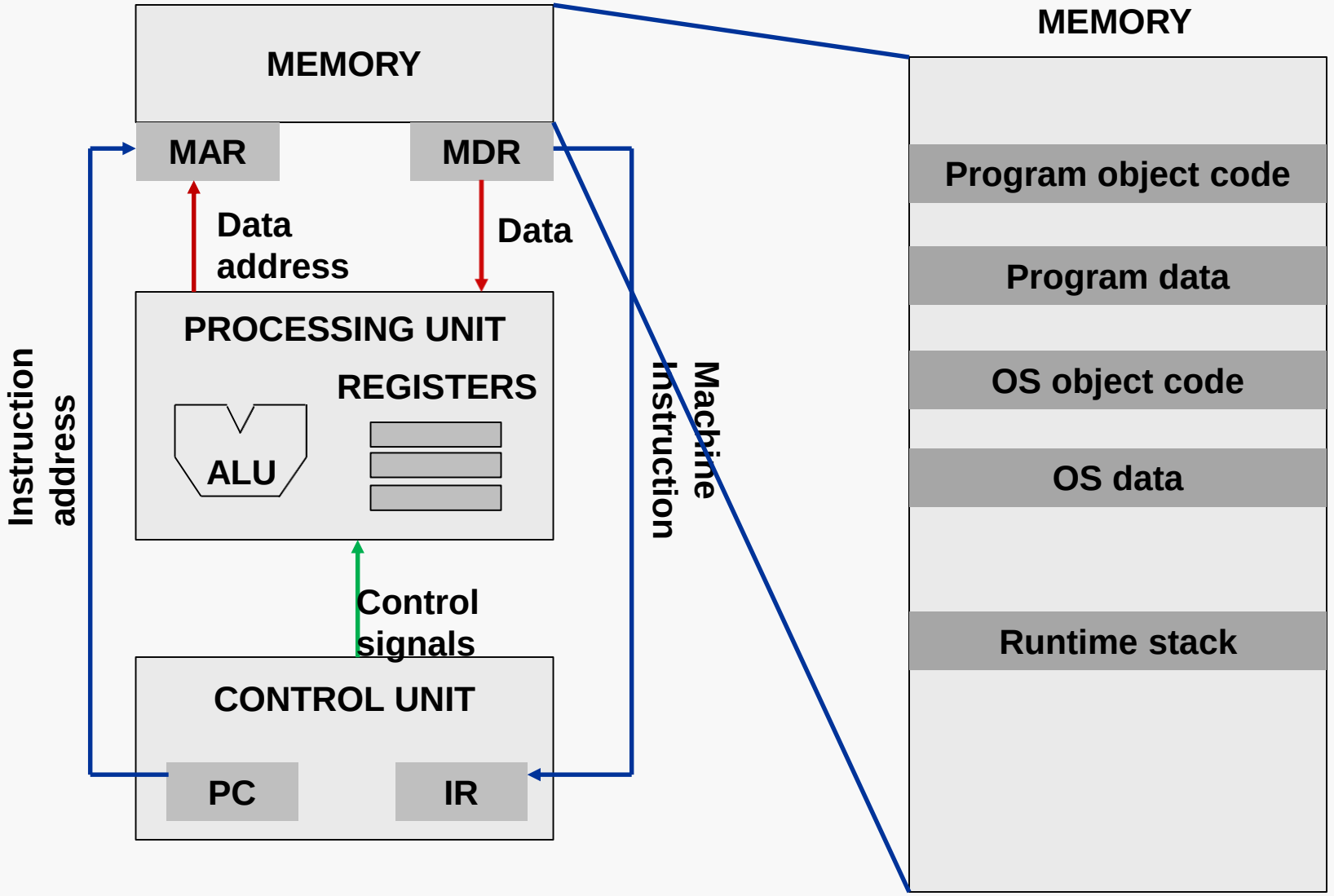
2004: Intel Announces EM64T extension to IA32

- 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

The von Neumann Machine



The von Neumann Machine

Programmer-visible State:

PC: *program counter*

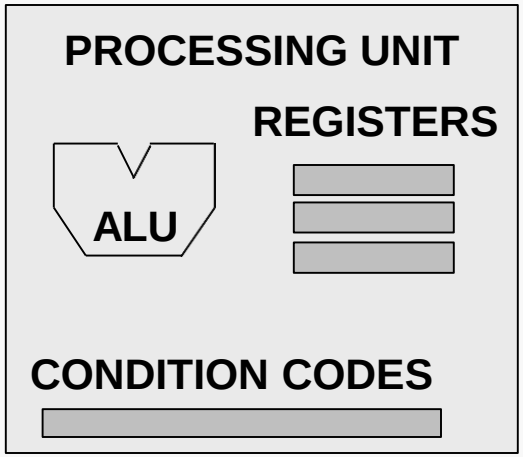
- stores address of next instruction
- called EIP (IA32) or RIP (x86-64)

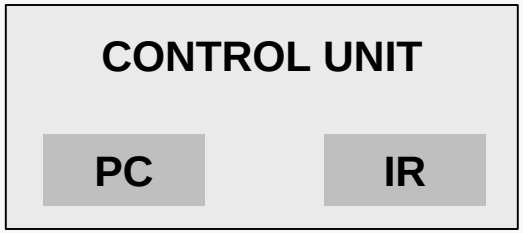
Register file:

- stores heavily used program data

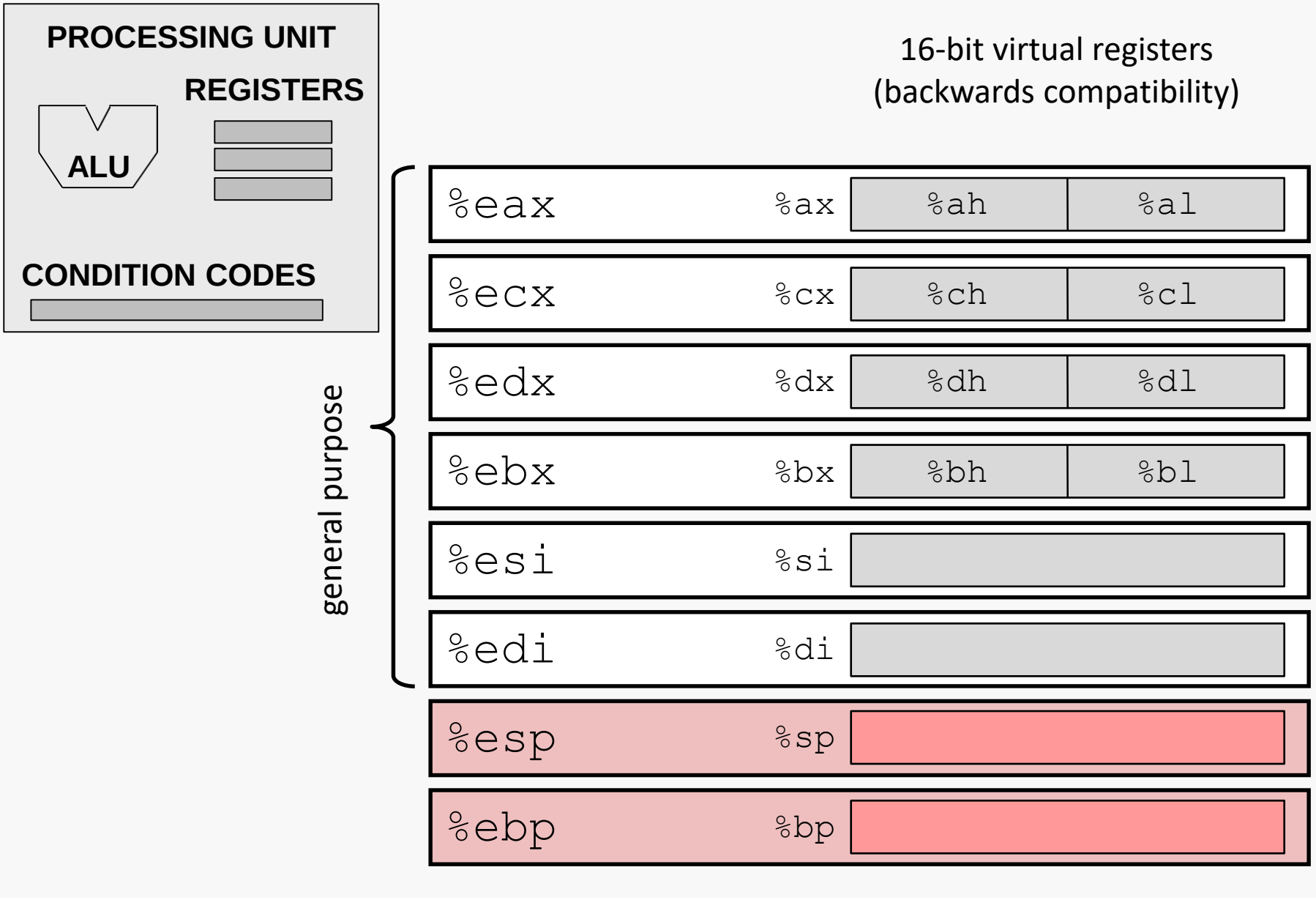
Condition codes

- store status information about most recent arithmetic-logical operation
- used for conditional branching





IA32 Integer Registers



x86-64 Integer Registers

%rax	%eax	%r8	%r8d
%rbx	%ebx	%r9	%r9d
%rcx	%ecx	%r10	%r10d
%rdx	%edx	%r11	%r11d
%rsi	%esi	%r12	%r12d
%rdi	%edi	%r13	%r13d
%rsp	%esp	%r14	%r14d
%rbp	%ebp	%r15	%r15d

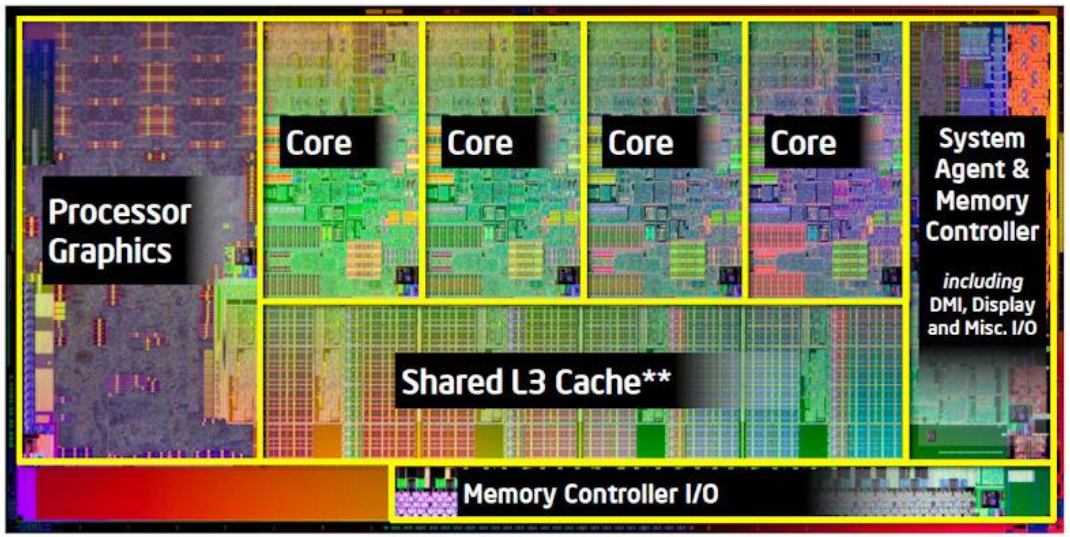
- Extend existing registers. Add 8 new ones.
- Make **%ebp/%rbp** general purpose

Programming the Machine

```
// C source 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 -m64 -std=c99 imax.c
```

```
457f464c0101000100000000000000000
0001000300010000000000000000000000
00bc000000000000340000000000028
000900068955be50845453b7c0c8b05
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:
        pushq   %rbp
        movq    %rsp, %rbp
        subq    $8, %rsp
        movl    %edi, -4(%rbp)
        movl    %esi, -8(%rbp)
        movl    -4(%rbp), %eax
        cmpl    -8(%rbp), %eax
        jl      .L4
        movl    -4(%rbp), %eax
        jmp     .L5
.L4:
        movl    -8(%rbp), %eax
.L5:
        popq    %rbp
        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.