Floating Point Puzzles

- For each of the following C expressions, either:
  - Argue that it is true for all argument values
  - Explain why not true

```
int x = …;

float f = …;

double d = …;
```

Assume neither `d` nor `f` is NaN

1.	`x == (int)(double) x`	**T**
2.	`x == (int)(float) x`	**F**
3.	`f == (float)(double) f`	**T**
4.	`d == (double)(float) d`	**F**
5.	`f == -(-f);`	**T**
6.	`1.0/2 == 1/2.0`	**T**
7.	`d×d >= 0.0`	**T**
8.	`(f+d)-f == d`	**F**

Summary

- **IEEE Floating Point has clear mathematical properties**
- **Represents numbers of form $M \times 2^E$**
- **One can reason about operations independent of implementation**
  - As if computed with perfect precision and then rounded
- **Not the same as real arithmetic**
  - Violates associativity/distributivity
  - Makes life difficult for compilers & serious numerical applications programmers

Intel x86 Evolution: Milestones

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

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

• 386	1985	275K	16-33

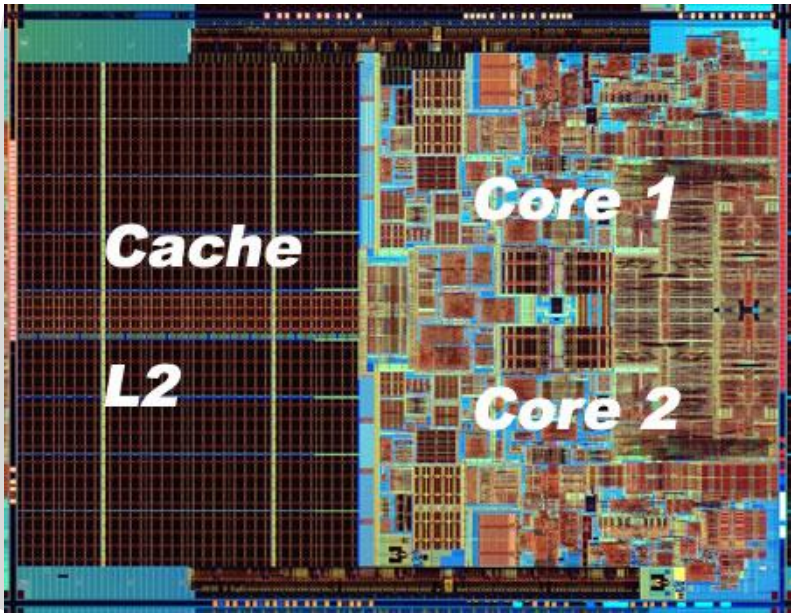
- First 32 bit processor , referred to as IA32
- Added “flat addressing”
- Capable of running Unix
- 32-bit Linux/gcc uses no instructions introduced in later models

• Pentium 4F	2005	230M	2800-3800

- First 64-bit processor
- Meanwhile, Pentium 4s (Netburst arch.) phased out in favor of “Core” line

Intel x86 Processors

- Machine Evolution
  - 486 1989 1.9M
  - Pentium 1993 3.1M
  - Pentium/MMX 1997 4.5M
  - PentiumPro 1995 6.5M
  - Pentium III 1999 8.2M
  - Pentium 4 2001 42M
  - Core 2 Duo 2006 291M



- Added Features
  - Instructions to support multimedia operations
    - Parallel operations on 1, 2, and 4-byte data, both integer & FP
  - Instructions to enable more efficient conditional operations
- Linux/GCC Evolution
  - Very limited

Name	*Date*	*Transistors*
• Itanium	2001	10M

- First shot at 64-bit architecture: first called IA64
- Radically new instruction set designed for high performance
- Can run existing IA32 programs
  - On-board “x86 engine”
- Joint project with Hewlett-Packard

• Itanium 2	2002	221M

- Big performance boost

• Itanium 2 Dual-Core	2006	1.7B

- Itanium has not taken off in marketplace
  - Lack of backward compatibility, no good compiler support, Pentium 4 got too good

x86 Clones: Advanced Micro Devices (AMD)

- Historically
  - AMD has followed just behind Intel
  - A little bit slower, a lot cheaper
- Then
  - Recruited top circuit designers from Digital Equipment Corp. and other downward trending companies
  - Built Opteron: tough competitor to Pentium 4
  - Developed x86-64, their own extension to 64 bits
- Recently
  - Intel much quicker with dual core design
  - Intel currently far ahead in performance
  - em64t backwards compatible to x86-64

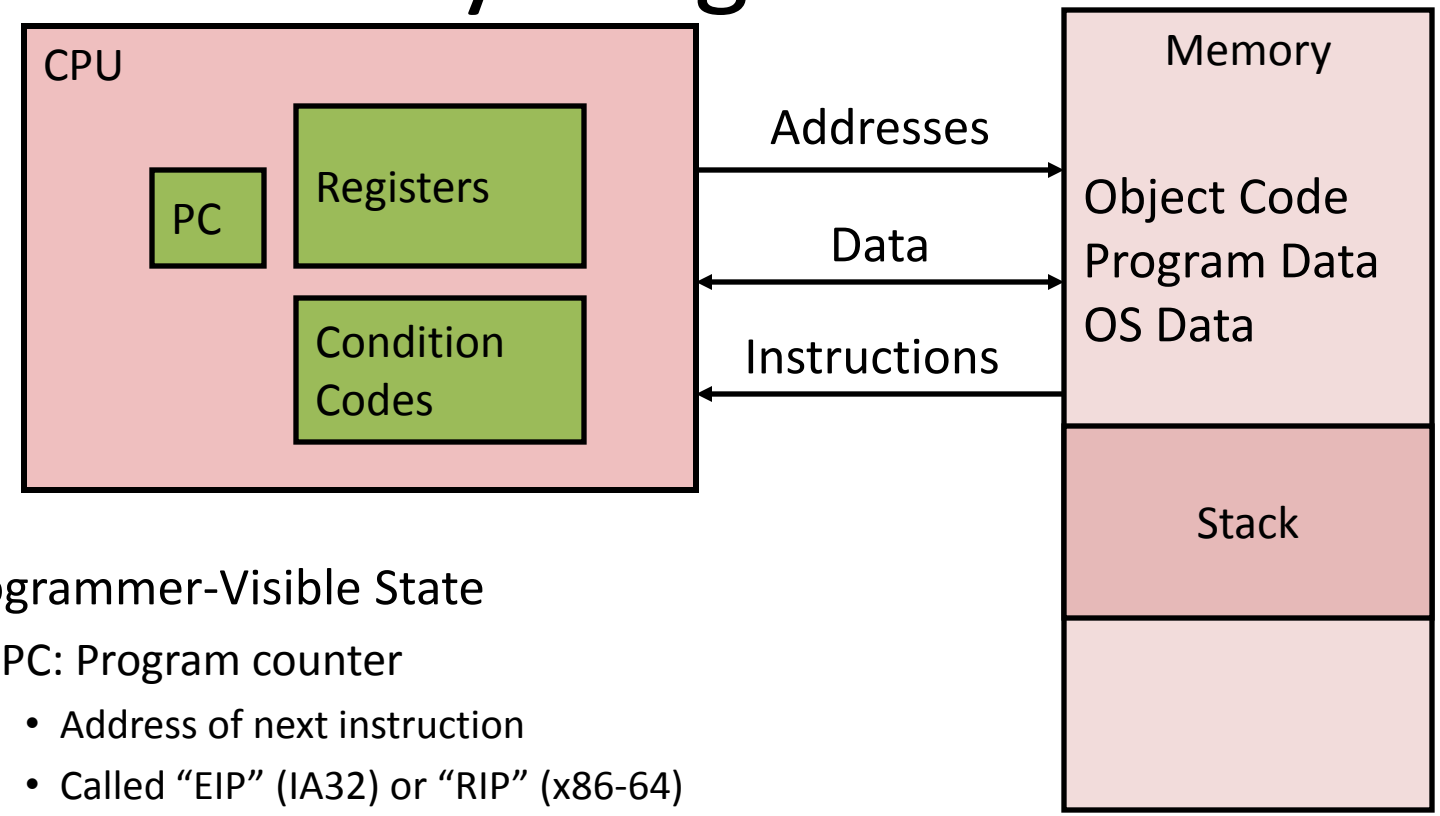
Intel’s 64 bit

- 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 is better
- 2004: Intel Announces EM64T extension to IA32
  - Extended Memory 64-bit Technology
  - Almost identical to x86-64!

- Architecture: (also instruction set architecture: ISA) The parts of a processor design that one needs to understand to write assembly code.
- Microarchitecture: Implementation of the architecture.

- Architecture examples: instruction set specification, registers.
- Microarchitecture examples: cache sizes and core frequency.

Assembly Programmer's View



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

Turning C into Object Code

– Code in files **`p1.c p2.c`**
– Compile with command: **`gcc -O p1.c p2.c -o p`**
  - Use optimizations (**`-O`**)
  - Put resulting binary in file **`p`**