MACHINE LEVEL PROGRAMMING 1: BASICS

CS 045

Computer Organization and Architecture

Prof. Donald J. Patterson

Adapted from Bryant and O'Hallaron, Computer Systems: A Programmer's Perspective, Third Edition

- HISTORY OF INTEL PROCESSORS AND ARCHITECTURES
- C, ASSEMBLY, MACHINE CODE
- ASSEMBLY BASICS: REGISTERS, OPERANDS, MOVE
- ARITHMETIC & LOGICAL OPERATIONS

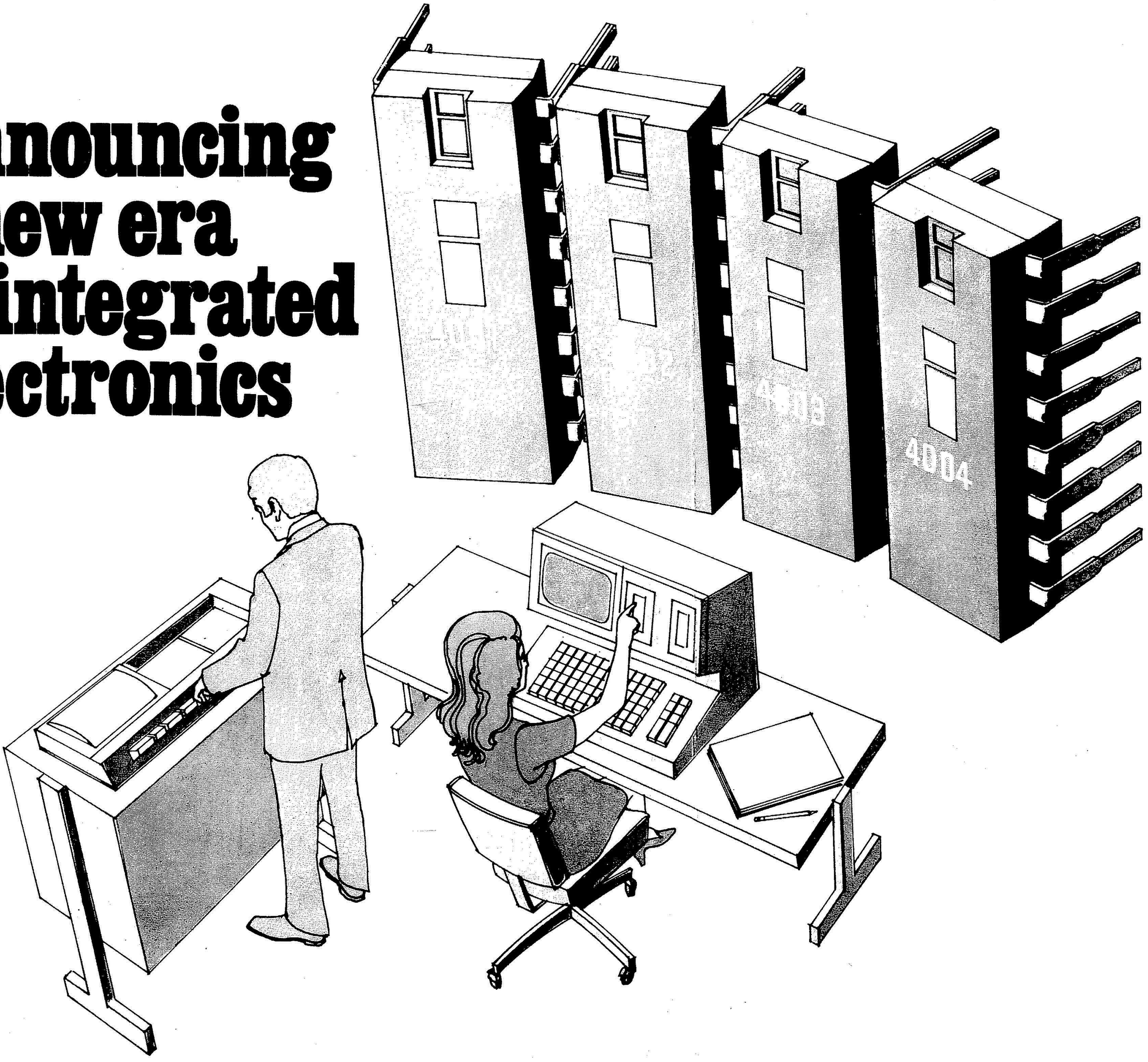
- HISTORY OF INTEL PROCESSORS AND ARCHITECTURES
- C, ASSEMBLY, MACHINE CODE
- ASSEMBLY BASICS: REGISTERS, OPERANDS, MOVE
- ARITHMETIC & LOGICAL OPERATIONS

HISTORY

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
- Documentation
  - https://software.intel.com/sites/default/files/managed/39/c5/325462-sdm-vol-1-2abcd-3abcd.pdf

Announcing a new era of integrated electronics



A micro-programmable computer on a chip!

Intel introduces an integrated CPU complete with a 4-bit parallel adder, sixteen 4-bit registers, an accumulator and a push-down stack on one chip. It's one of a family of four new ICs which comprise the MCS-4 micro computer system—the first system to bring you the power and flexibility of a dedicated general-purpose computer at low cost in as few as two dual in-line packages.

MCS-4 systems provide complete computing and control functions for test systems, data terminals, billing machines, measuring systems, numeric control systems and process control systems.

The heart of any MCS-4 system is a Type 4004 CPU, which includes a powerful set of 45 instructions. Adding one or more Type 4001 ROMs for program storage and data tables gives you a fully functioning micro-programmed computer. To this you may add Type 4002 RAMs for read-write memory and Type 4003 registers to expand the output ports.

Using no circuitry other than ICs from this family of four, you can create a system with 4096 8-bit bytes of ROM storage and 5120 bits of RAM storage. When you require rapid turn-around or need only a few systems, Intel's erasable and re-programmable ROM, Type 1701, may be substituted for the Type 4001 mask-programmed ROM.

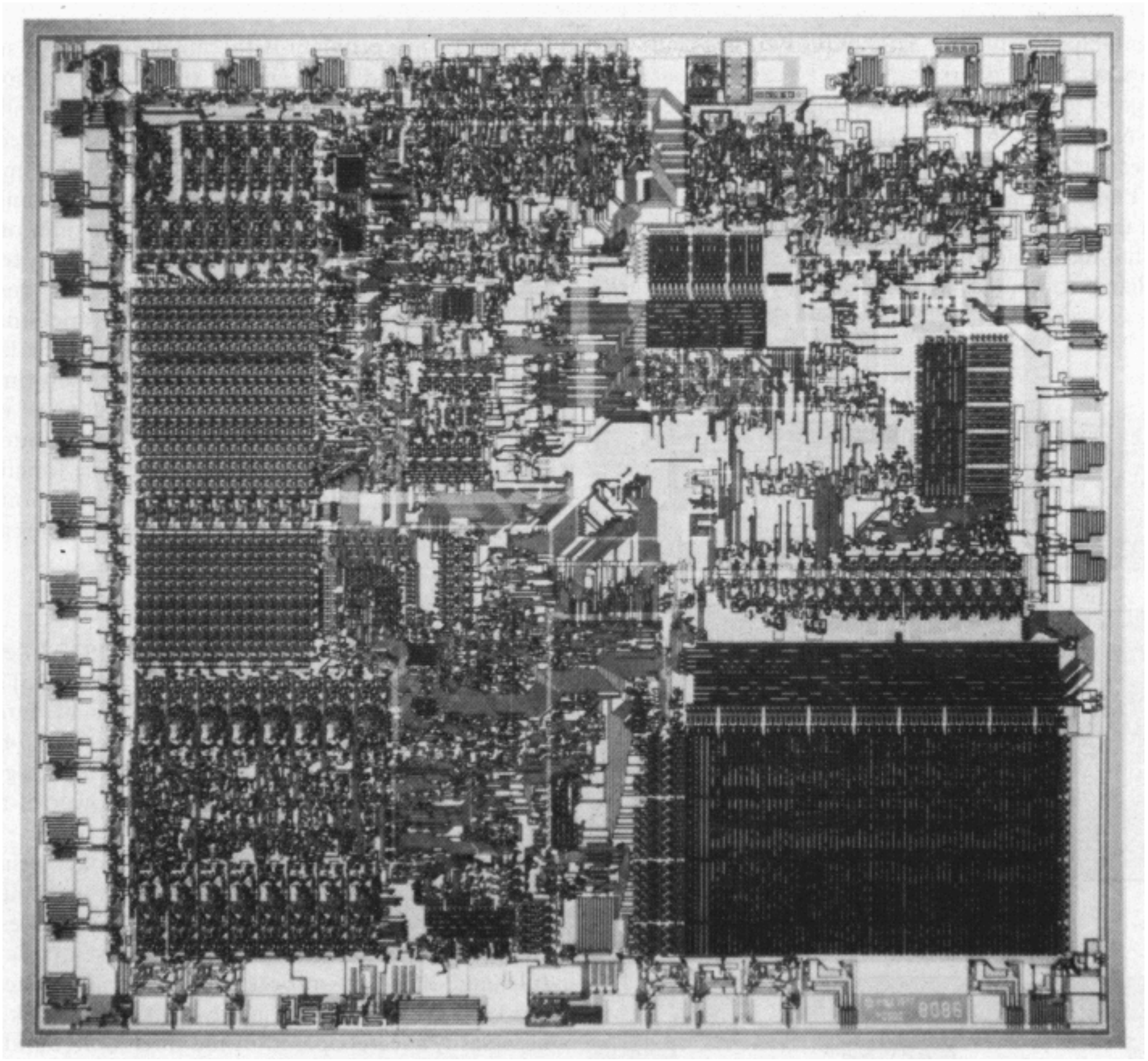
MCS-4 systems interface easily with switches, key-boards, displays, teletypewriters, printers, readers, A-D converters and other popular peripherals.

The MCS-4 family is now in stock at Intel's Santa Clara headquarters and at our marketing headquarters in Europe and Japan. In the U.S., contact your local Intel representative for technical information and literature. In Europe, contact Intel at Avenue Louise 216, B 1050 Bruxelles, Belgium. Phone 492003. In Japan, contact Intel Japan, Inc., Parkside Flat Bldg. No. 4-2-2, Sendagaya, Shibuya-Ku, Tokyo 151. Phone 03-403-4747.

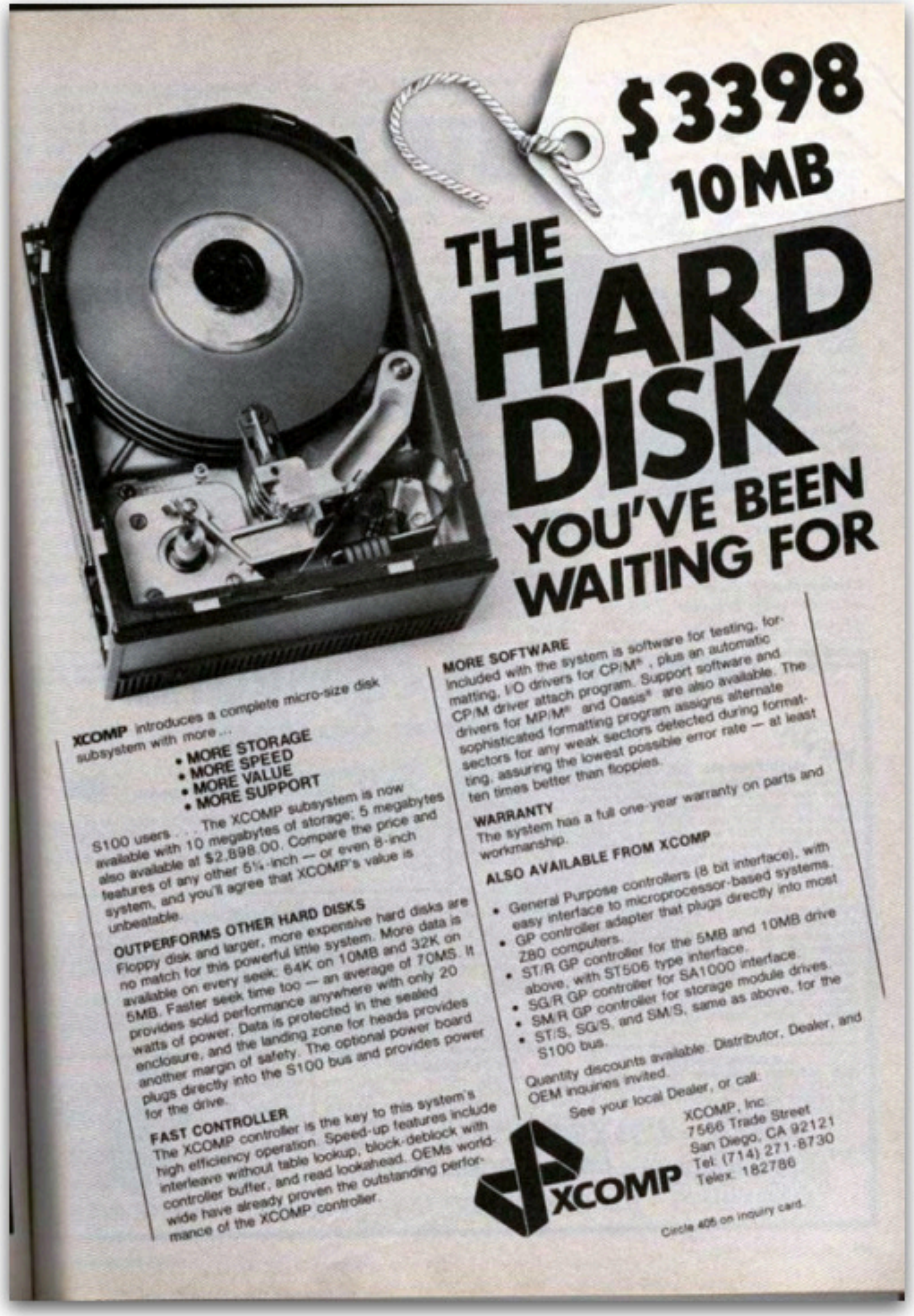
Intel Corporation now produces micro computers, memory devices and memory systems at 3065 Bowers Avenue, Santa Clara, Calif. 95051. Phone (408) 246-7501.

intel® delivers.

ELECTRONICS NEWS, 11/15/71



- 1978: By 1978, microprocessor chip complexity had increased an order of magnitude over the first four-bit processors. Intel’s 8086 16-bit processor, shown here, integrates 29,000 transistors, over 12 times as many as the original Intel 4004. [citation]



$3398
10MB

THE HARD DISK YOU'VE BEEN WAITING FOR

XCOMP introduces a complete micro-size disk subsystem with more...

- MORE STORAGE
- MORE SPEED
- MORE VALUE
- MORE SUPPORT

S100 users . . . The XCOMP subsystem is now available with 10 megabytes of storage; 5 megabytes also available at $2,898.00. Compare the price and features of any other 5¼-inch — or even 8-inch system, and you'll agree that XCOMP's value is unbeatable.

OUTPERFORMS OTHER HARD DISKS
Floppy disk and larger, more expensive hard disks are no match for this powerful little system. More data is available on every seek: 64K on 10MB and 32K on 5MB. Faster seek time too — an average of 70MS. It provides solid performance anywhere with only 20 watts of power. Data is protected in the sealed enclosure, and the landing zone for heads provides another margin of safety. The optional power board plugs directly into the S100 bus and provides power for the drive.

FAST CONTROLLER
The XCOMP controller is the key to this system's high efficiency operation. Speed-up features include interleave without table lookup, block-deblock with controller buffer, and read lookahead. OEMs world-wide have already proven the outstanding performance of the XCOMP controller.

MORE SOFTWARE
Included with the system is software for testing, formatting, I/O drivers for CP/M®, plus an automatic CP/M driver attach program. Support software and drivers for MP/M® and Oasis® are also available. The sophisticated formatting program assigns alternate sectors for any weak sectors detected during formatting, assuring the lowest possible error rate — at least ten times better than floppies.

WARRANTY
The system has a full one-year warranty on parts and workmanship.

ALSO AVAILABLE FROM XCOMP

- General Purpose controllers (8 bit interface), with easy interface to microprocessor-based systems.
- GP controller adapter that plugs directly into most Z80 computers.
- ST/R GP controller for the 5MB and 10MB drive above, with ST506 type interface.
- SG/R GP controller for SA1000 interface.
- SM/R GP controller for storage module drives.
- ST/S, SG/S, and SM/S, same as above, for the S100 bus.

Quantity discounts available. Distributor, Dealer, and OEM inquiries invited.

See your local Dealer, or call:

XCOMP

XCOMP, Inc.
7566 Trade Street
San Diego, CA 92121
Tel: (714) 271-8730
Telex: 182786

Circle 405 on inquiry card.

HISTORY

INTEL X86 PROCESSORS

- 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) (e.g. ARM)

- But, Intel has done just that!
  - In terms of speed. Less so for low power.

HISTORY
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 4E2004 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
  - i7-7700 2017 ? 4500

HISTORY

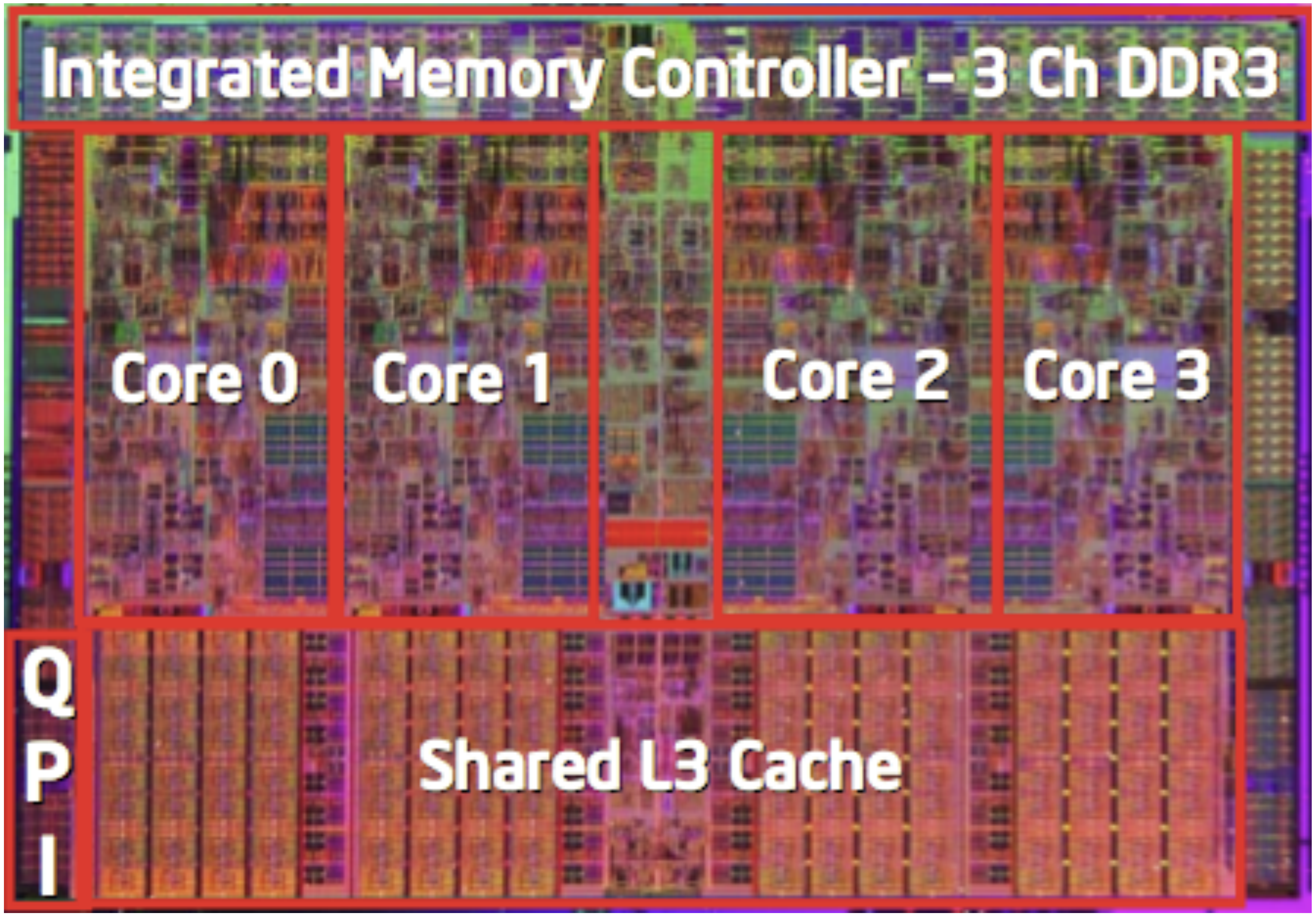
INTEL X86 EVOLUTION

- Machine Evolution

386	1985	0.3M
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
Core i7	2008	731M

  - Added Features
  - Instructions to support multimedia operations
  - Instructions to enable more efficient conditional operations

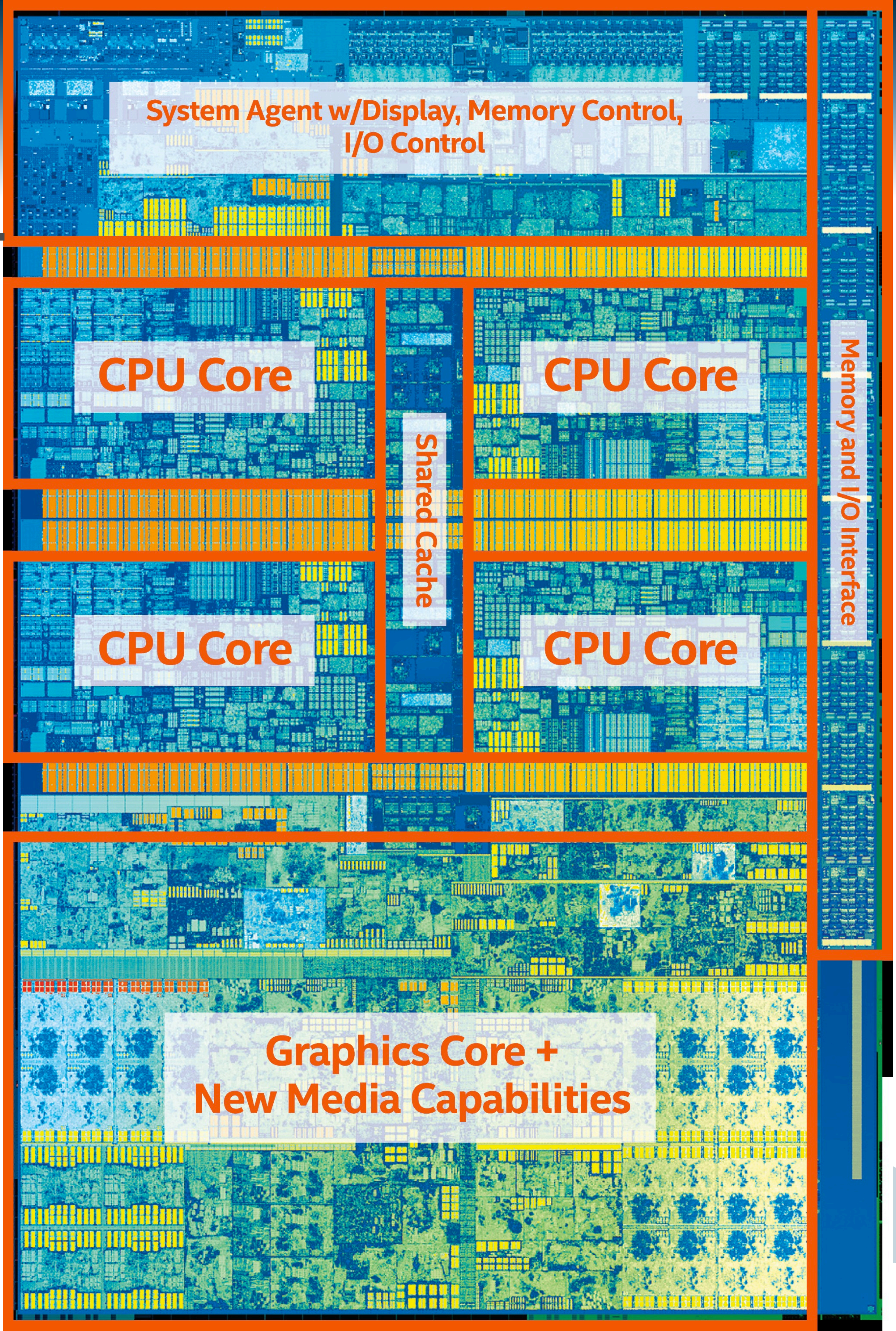
INTEL X86 EVOLUTION



HISTORY

2017 STATE OF THE ART

- Core i7 Kaby Lake 2017
- Desktop Model
  - 4 cores
  - Integrated 4k Video
  - 2.9-4.5 GHz
  - 65W


System Agent w/Display, Memory Control, I/O Control
CPU Core
CPU Core
Shared Cache
CPU Core
CPU Core
Memory and I/O Interface
Graphics Core + New Media Capabilities

HISTORY

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
- Recent Years
  - Intel got its act together
    - Leads the world in semiconductor technology
  - AMD has fallen behind
    - Relies on external semiconductor manufacturer

HISTORY

INTEL 64 BIT HISTORY

- 2001: Intel Attempts Radical Shift from IA32 to IA64
  - Totally different architecture (Itanium)
  - Executes IA32 code only as legacy
  - Performance disappointing
- 2003: AMD Steps 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!
- All but low-end x86 processors support x86-64
  - But, lots of code still runs in 32-bit mode

- HISTORY OF INTEL PROCESSORS AND ARCHITECTURES
- C, ASSEMBLY, MACHINE CODE
- ASSEMBLY BASICS: REGISTERS, OPERANDS, MOVE
- ARITHMETIC & LOGICAL OPERATIONS

- HISTORY OF INTEL PROCESSORS AND ARCHITECTURES
- C, ASSEMBLY, MACHINE CODE
- ASSEMBLY BASICS: REGISTERS, OPERANDS, MOVE
- ARITHMETIC & LOGICAL OPERATIONS

C, ASSEMBLY, MACHINE CODE

DEFINITIONS

- Architecture: (also ISA: instruction set architecture) The parts of a processor design that one needs to understand to write assembly/machine code.
  - Examples: instruction set specification, registers.
- Microarchitecture: Implementation of the architecture.
  - Examples: cache sizes and core frequency.
- Code Forms:
  - Machine Code: The byte-level programs that a processor executes
  - Assembly Code: A text representation of machine code

- Example ISAs:
  - Intel: x86, IA32, Itanium, x86-64
  - ARM: Used in almost all mobile phones

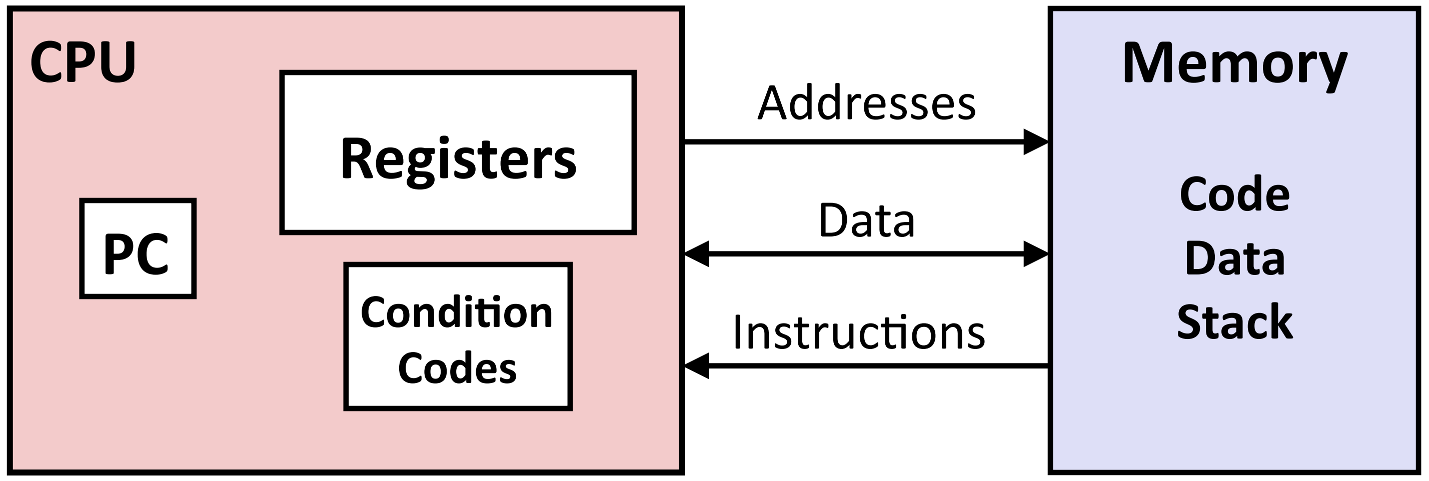
C, ASSEMBLY, MACHINE CODE

DEFINITIONS

- Architecture: (also ISA: instruction set architecture) The parts of a processor design that one needs to understand or write assembly/machine code.
  - Examples: instruction set specification, registers.
- Microarchitecture: Implementation of the architecture.
  - Examples: cache sizes and core frequency.
- Code Forms:
  - Machine Code: The byte-level programs that a processor executes
  - Assembly Code: A text representation of machine code

- Example ISAs:
  - Intel: x86, IA32, Itanium, x86-64
  - ARM: Used in almost all mobile phones

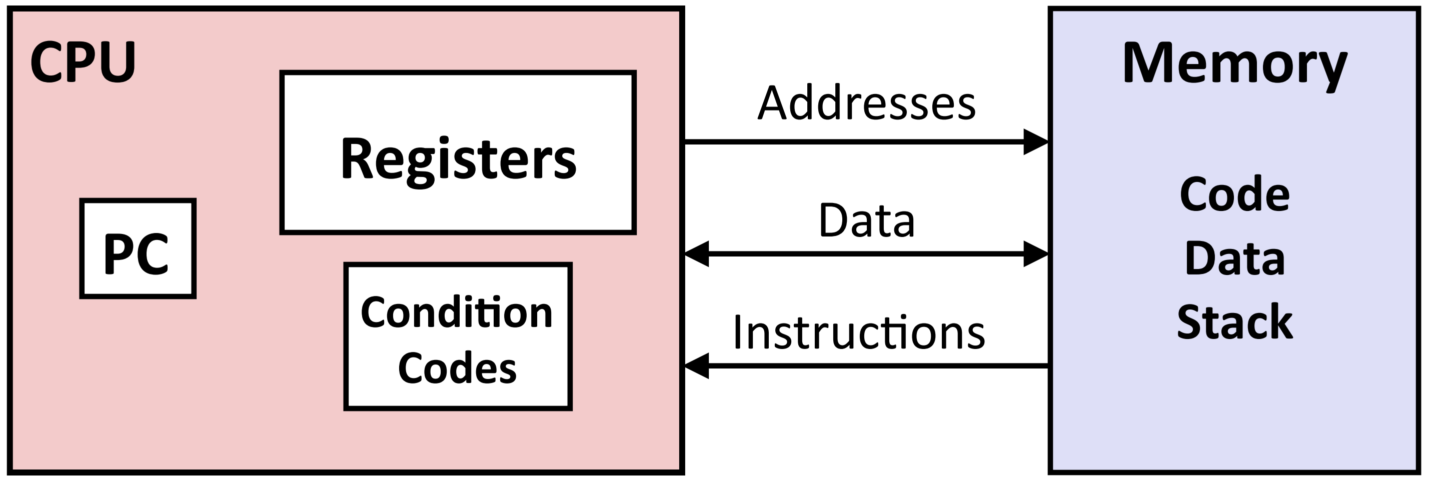
ASSEMBLY/MACHINE CODE VIEW



- Programmer-Visible State
  - PC: Program counter
    - Address of next instruction
    - Called “RIP” (x86-64)
  - Register file
    - Heavily used program data
  - Condition codes
    - Store status information about most recent arithmetic or logical operation
    - Used for conditional branching

C, ASSEMBLY, MACHINE CODE

ASSEMBLY/MACHINE CODE VIEW

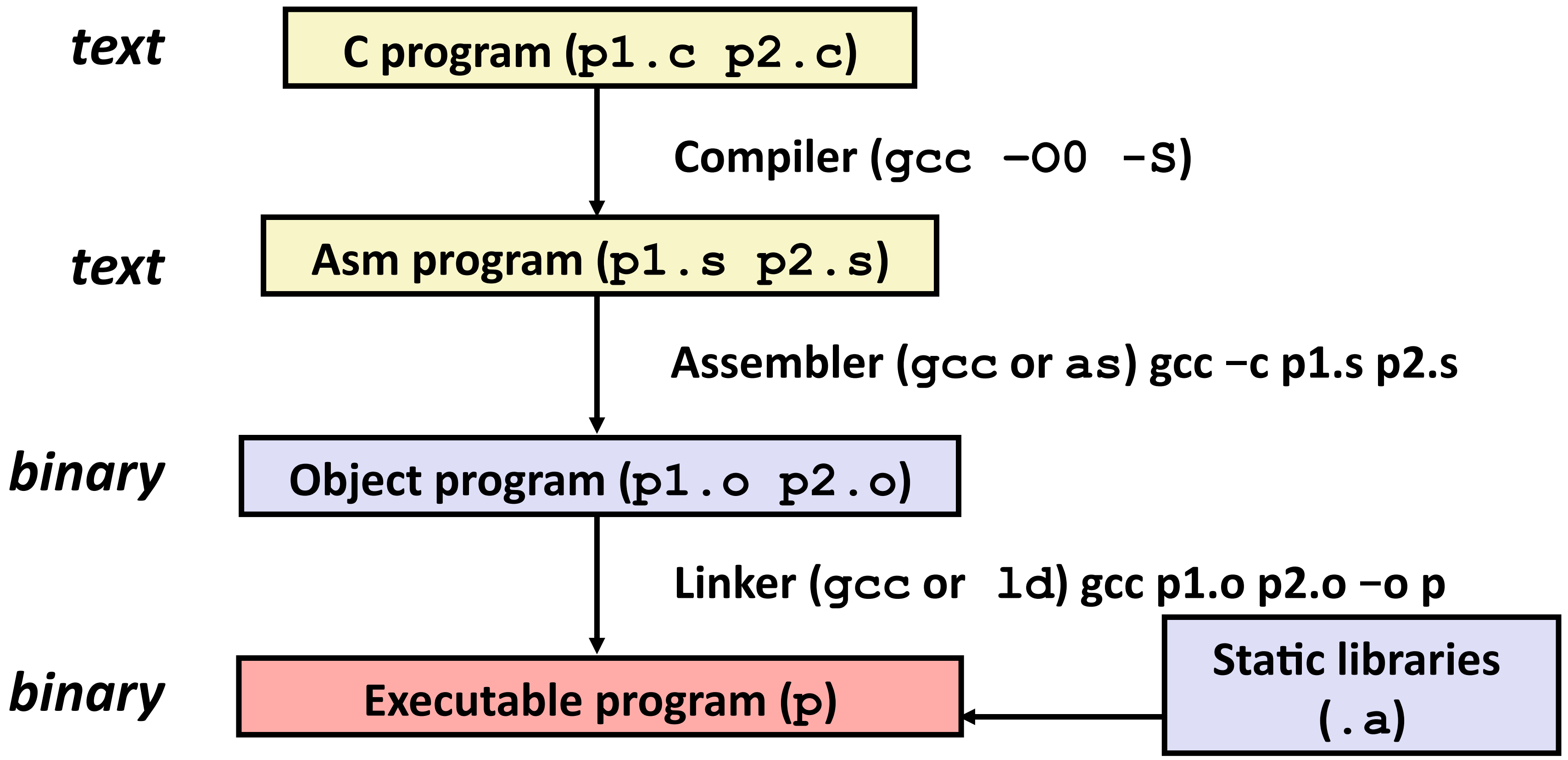


- Memory
  - Byte addressable array
  - Code and user data
  - Stack to support procedures

C, ASSEMBLY, MACHINE CODE

TURNING C INTO OBJECT CODE

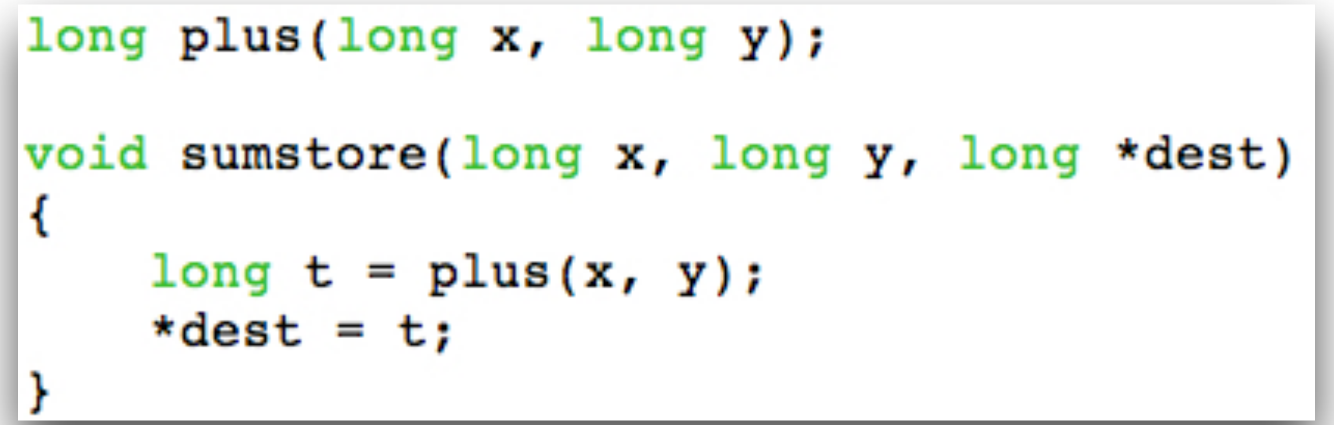
- Code in files `p1.c p2.c`
  - Compile with command: `gcc –O0 p1.c p2.c -o p`
    - `-O0` use default optimizations
    - `-Og` [New to recent versions of GCC] "basic optimizations"
    - Put resulting binary in file p



C, ASSEMBLY, MACHINE CODE

COMPILING INTO ASSEMBLY

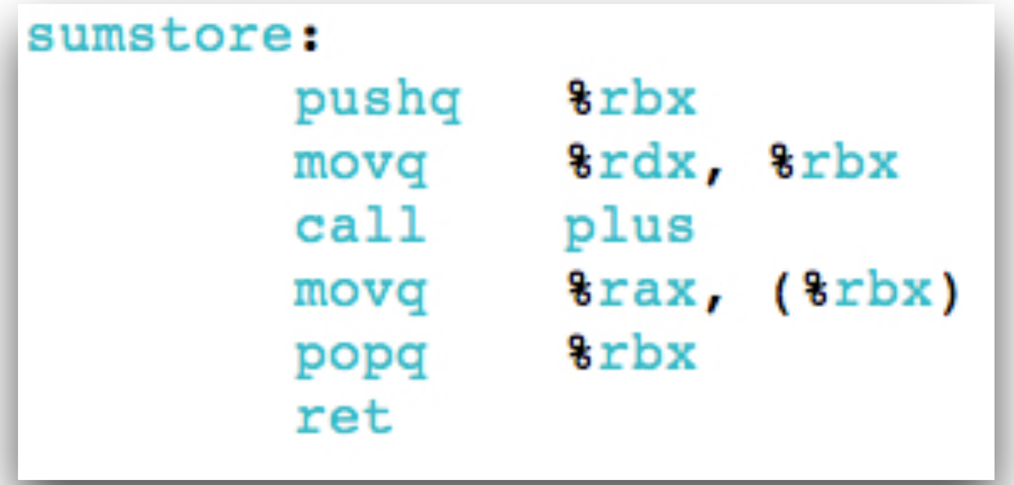
`sum.c`



```c
long plus(long x, long y);

void sumstore(long x, long y, long *dest)
{
    long t = plus(x, y);
    *dest = t;
}
```

`sum.s`



```
sumstore:
        pushq   %rbx
        movq    %rdx, %rbx
        call    plus
        movq    %rax, (%rbx)
        popq    %rbx
        ret
```

- On wcpkneel
  - Compile with command: `gcc -O1 -S sum.c`
  - this produces sum.s

- Note: Will get very different results on different machine (Linux, Mac OS-X, …) due to different versions of gcc and different compiler settings.

ASSEMBLY CHARACTERISTICS: DATA TYPES

- "Integer" data of 1, 2, 4, or 8 bytes
  - Data values
  - Addresses (untyped pointers)
- Floating point data of 4, 8, or 10 bytes
- Code
  - Byte sequences encoding series of instructions
- No aggregate types such as arrays or structures
  - Just contiguously allocated bytes in memory

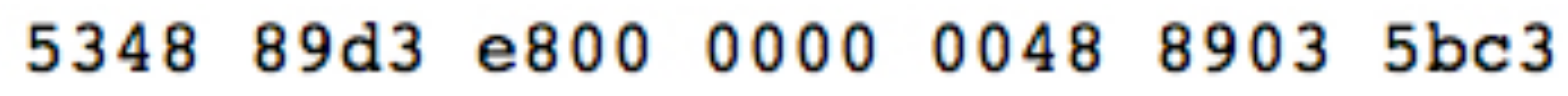
ASSEMBLY CHARACTERISTICS: OPERATIONS

- Perform arithmetic function on register or memory data
- Transfer data between memory and register
  - Load data from memory into register
  - Store register data into memory
- Transfer control
  - Unconditional jumps to/from procedures
  - Conditional branches

C, ASSEMBLY, MACHINE CODE

OBJECT CODE

```
sumstore as machine language
```



```
5348 89d3 e800 0000 0048 8903 5bc3
```

C, ASSEMBLY, MACHINE CODE

OBJECT CODE

```
5348 89d3 e800 0000 0048 8903 5bc3
```

Code for `sumstore`

```
0x0400595:
    0x53
    0x48
    0x89
    0xd3
    0xe8
    0xf2
    0xff
    0xff
    0xff
    0x48
    0x89
    0x03
    0x5b
    0xc3
```

- **Total of 14 bytes**
- **Each instruction 1, 3, or 5 bytes**
- **Starts at address `0x0400595`**

- Assembler
  - Translates .s into .o
  - Binary encoding of each instruction
  - Nearly-complete image of executable code
  - Missing linkages between code in different files
- Linker
  - Resolves references between files
  - Combines with static run-time libraries
    - E.g., code for malloc, printf
  - Some libraries are dynamically linked
    - Linking occurs when program begins execution

C, ASSEMBLY, MACHINE CODE

MACHINE INSTRUCTION EXAMPLETRANSLATES .S

```
*dest = t;
```

```
movq %rax, (%rbx)
```

```
00 0000 0048 8903 5bc3
```

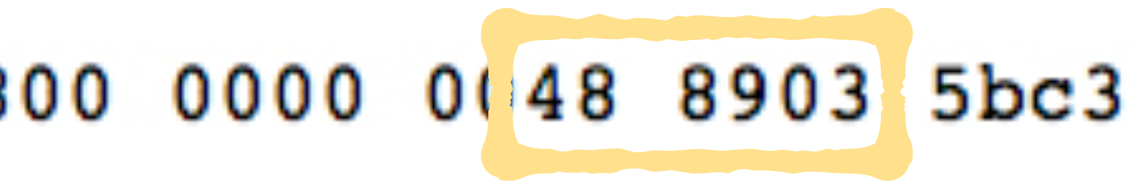
- C Code
  - Store value t where designated by dest
- Assembly
  - Move 8-byte value to memory
  - Quad words in x86-64 parlance
- Operands:
  - t: Register %rax
  - dest: Register %rbx
  - *dest: MemoryM[%rbx]
- Object Code
  - 3-byte instruction
  - 0x48 89 03

C, ASSEMBLY, MACHINE CODE

MACHINE INSTRUCTION EXAMPLETRANSLATES .S

```
*dest = t;
```

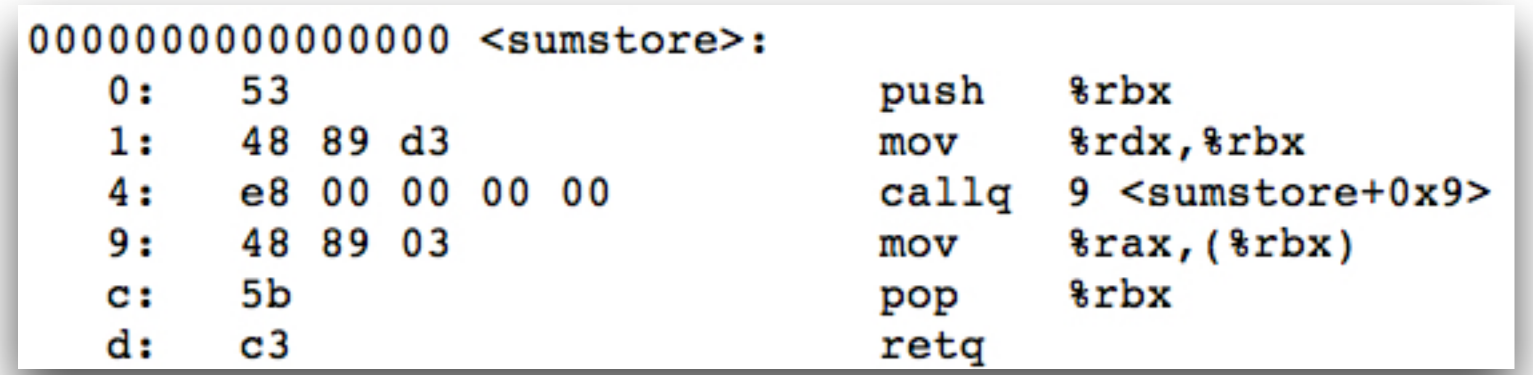
```
movq %rax, (%rbx)
```



```
00  0000  0048  8903  5bc3
```

- C Code
  - Store value t where designated by dest
- Assembly
  - Move 8-byte value to memory
  - Quad words in x86-64 parlance
- Operands:
  - t: Register %rax
  - dest: Register %rbx
  - *dest: MemoryM[%rbx]
- Object Code
  - 3-byte instruction
  - 0x48 89 03

DISASSEMBLING OBJECT CODE

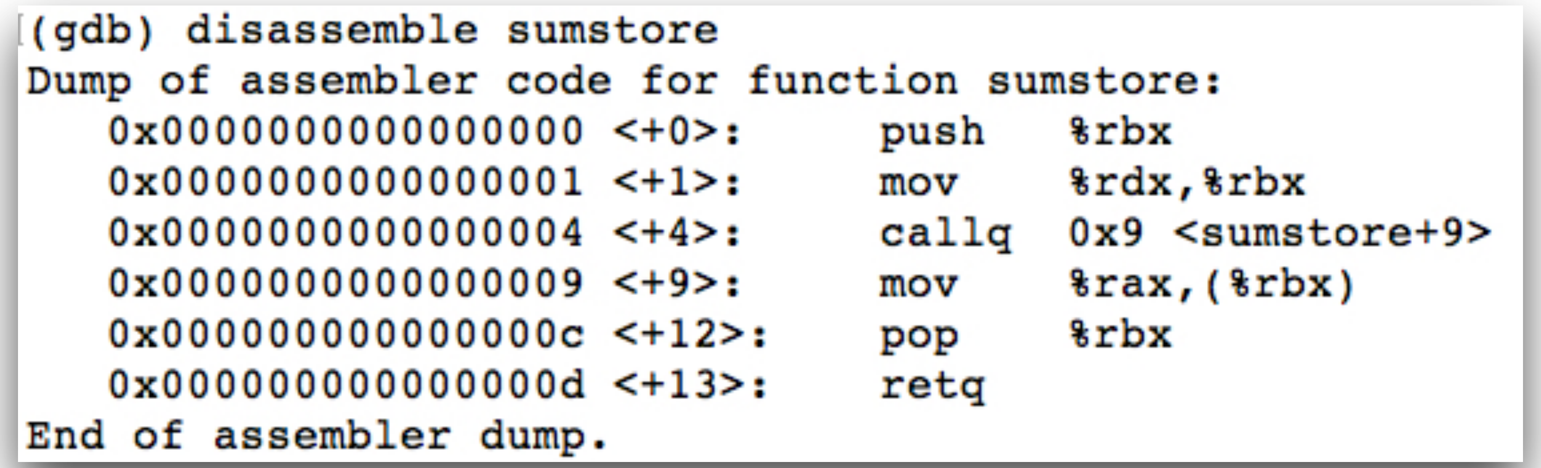


```
0000000000000000 <sumstore>:
   0:    53                       push   %rbx
   1:    48 89 d3                 mov    %rdx,%rbx
   4:    e8 00 00 00 00           callq  9 <sumstore+0x9>
   9:    48 89 03                 mov    %rax,(%rbx)
   c:    5b                       pop    %rbx
   d:    c3                       retq
```

- Disassembler
  - `objdump -d sum`
    - Useful tool for examining object code
    - Analyzes bit pattern of series of instructions
    - Produces approximate rendition of assembly code
    - Can be run on either a.out (complete executable) or .o file

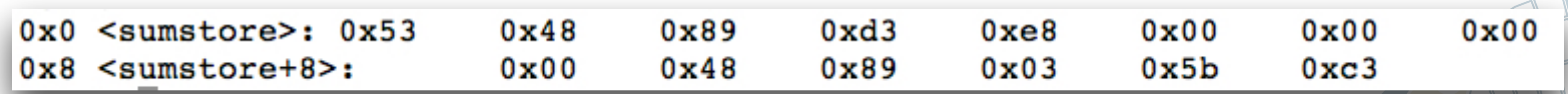
C, ASSEMBLY, MACHINE CODE

ALTERNATE DISASSEMBLY



```
(gdb) disassemble sumstore
Dump of assembler code for function sumstore:
   0x0000000000000000 <+0>:     push   %rbx
   0x0000000000000001 <+1>:     mov    %rdx,%rbx
   0x0000000000000004 <+4>:     callq  0x9 <sumstore+9>
   0x0000000000000009 <+9>:     mov    %rax,(%rbx)
   0x000000000000000c <+12>:    pop    %rbx
   0x000000000000000d <+13>:    retq
End of assembler dump.
```

- Within gdb Debugger
  - `gdb sum`
  - `disassemble sumstore`
    - Disassemble procedure
  - `x/14xb sumstore`
    - Examine the 14 bytes starting at sumstore



```
0x0 <sumstore>: 0x53    0x48    0x89    0xd3    0xe8    0x00    0x00    0x00
0x8 <sumstore+8>:       0x00    0x48    0x89    0x03    0x5b    0xc3
```

C, ASSEMBLY, MACHINE CODE

WHAT CAN BE DISASSEMBLED?

```
% objdump -d WINWORD.EXE

WINWORD.EXE:   file format pei-i386

No symbols in "WINWORD.EXE".
Disassembly of section .text:

30001000 <.text>:
30001000:
30001001:
30001003:
30001005:
3000100a:
```

Reverse engineering forbidden by Microsoft End User License Agreement

- Anything that can be interpreted as executable code
- Disassembler examines bytes and reconstructs assembly source

WESTMONT INSPIRED

COMPUTING LAB