16.480/552 Micro II and Embedded Systems Design: Introduction

Lecture 1: Introduction to Embedded System Design

Revised based on "Embedded Systems Design: A Unified Hardware/Software" by Vahid/Givargis
And "*The 8088 and 8086 Microprocessors*" by Triebel and Singh

Outline

- Embedded systems overview
  - What are they?
- Design challenge – optimizing design metrics
- Technologies
  - Processor technologies
  - IC technologies
  - Design technologies
- Introduction to 8088/8086

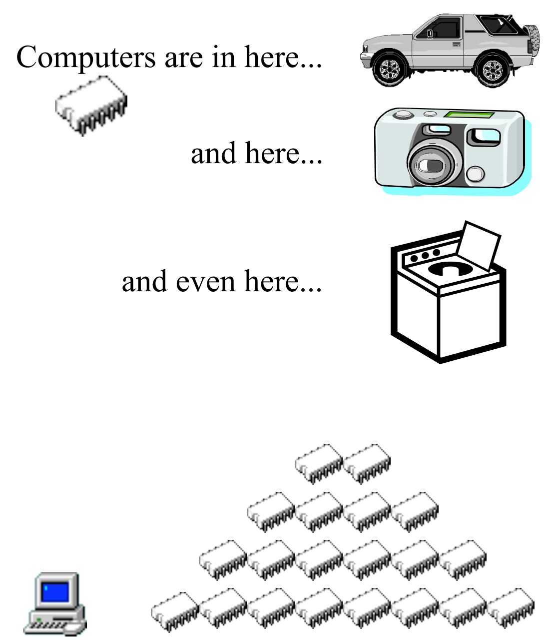
Embedded systems overview

- Computing systems are everywhere
- Most of us think of "desktop" computers
  - PC's
  - Laptops
  - Mainframes
  - Servers

- But there's another type of computing system
  - Far more common...

Embedded systems overview

- Embedded computing systems
  - Computing systems embedded within electronic devices
  - Hard to define. Nearly any computing system other than a desktop computer
  - Billions of units produced yearly, versus millions of desktop units
  - Perhaps 50 per household and per automobile

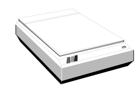
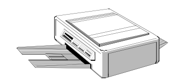
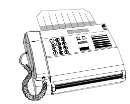
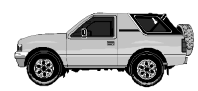
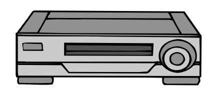
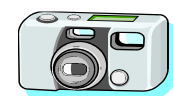
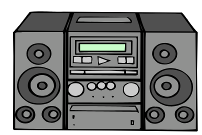


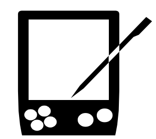
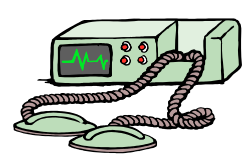
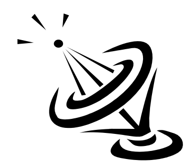
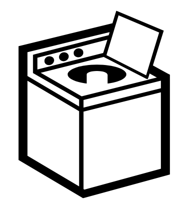
Lots more of these, though they cost a lot less each.

A "short list" of embedded systems

Anti-lock brakes
Auto-focus cameras
Automatic teller machines
Automatic toll systems
Automatic transmission
Avionic systems
Battery chargers
Camcorders
Cell phones
Cell-phone base stations
Cordless phones
Cruise control
Curbside check-in systems
Digital cameras
Disk drives
Electronic card readers
Electronic instruments
Electronic toys/games
Factory control
Fax machines
Fingerprint identifiers
Home security systems
Life-support systems
Medical testing systems

Modems
MPEG decoders
Network cards
Network switches/routers
On-board navigation
Pagers
Photocopiers
Point-of-sale systems
Portable video games
Printers
Satellite phones
Scanners
Smart ovens/dishwashers
Speech recognizers
Stereo systems
Teleconferencing systems
Televisions
Temperature controllers
Theft tracking systems
TV set-top boxes
VCR's, DVD players
Video game consoles
Video phones
Washers and dryers

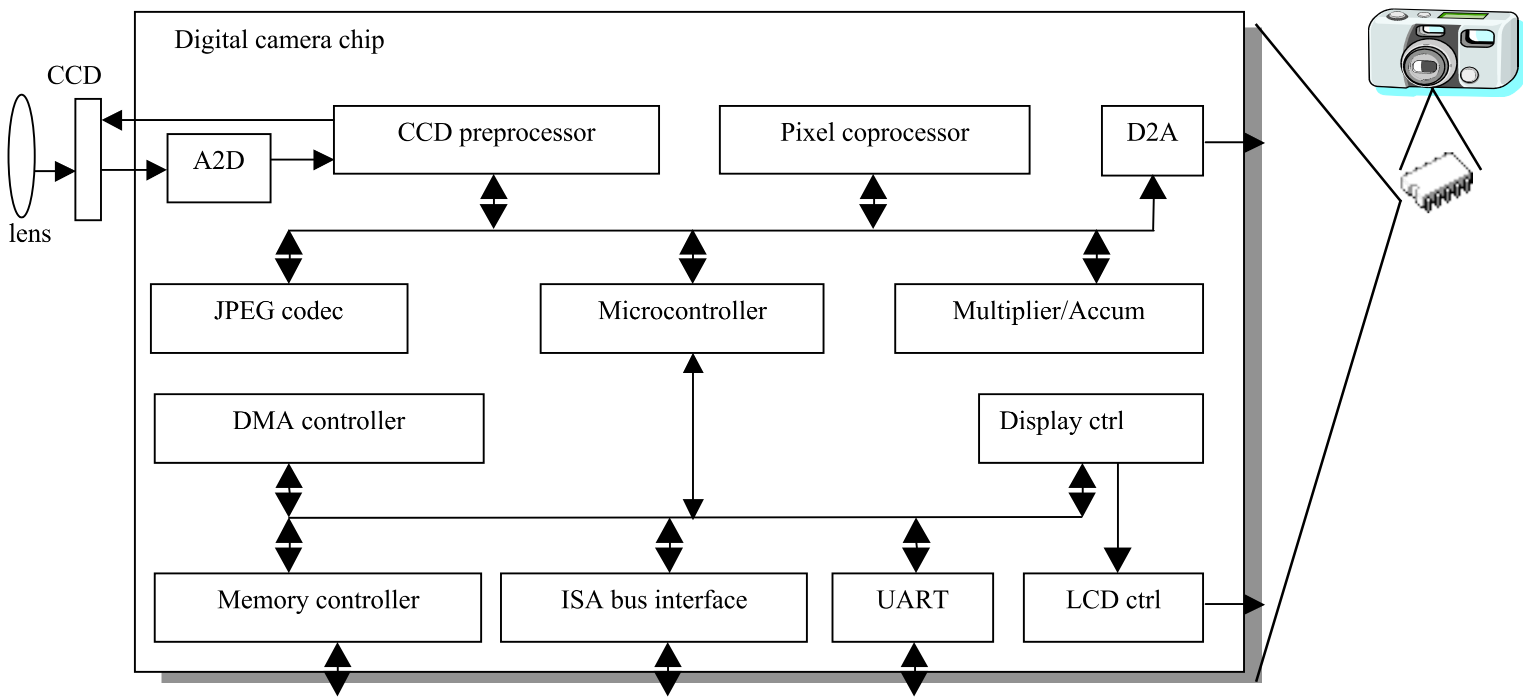





And the list goes on and on

Some common characteristics of embedded systems

- Single-functioned
  - Executes a single program, repeatedly
- Tightly-constrained
  - Low cost, low power, small, fast, etc.
- Reactive and real-time
  - Continually reacts to changes in the system's environment
  - Must compute certain results in real-time without delay

An embedded system example -- a digital camera



- Single-functioned -- always a digital camera
- Tightly-constrained -- Low cost, low power, small, fast
- Reactive and real-time -- only to a small extent

Design challenge – optimizing design metrics

- Obvious design goal:
  - Construct an implementation with desired functionality
- Key design challenge:
  - Simultaneously optimize numerous design metrics
- Design metric
  - A measurable feature of a system's implementation
  - Optimizing design metrics is a key challenge

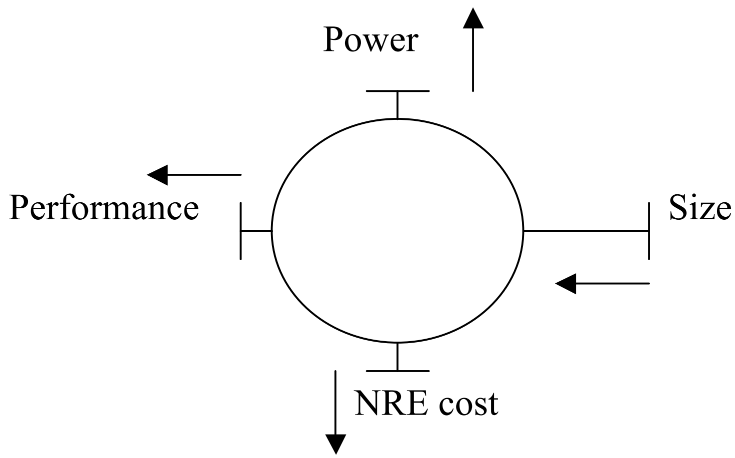
Design challenge – optimizing design metrics

- Common metrics
  - Unit cost: the monetary cost of manufacturing each copy of the system, excluding NRE cost
  - NRE cost (Non-Recurring Engineering cost): The one-time monetary cost of designing the system
  - Size: the physical space required by the system
  - Performance: the execution time or throughput of the system
  - Power: the amount of power consumed by the system
  - Flexibility: the ability to change the functionality of the system without incurring heavy NRE cost

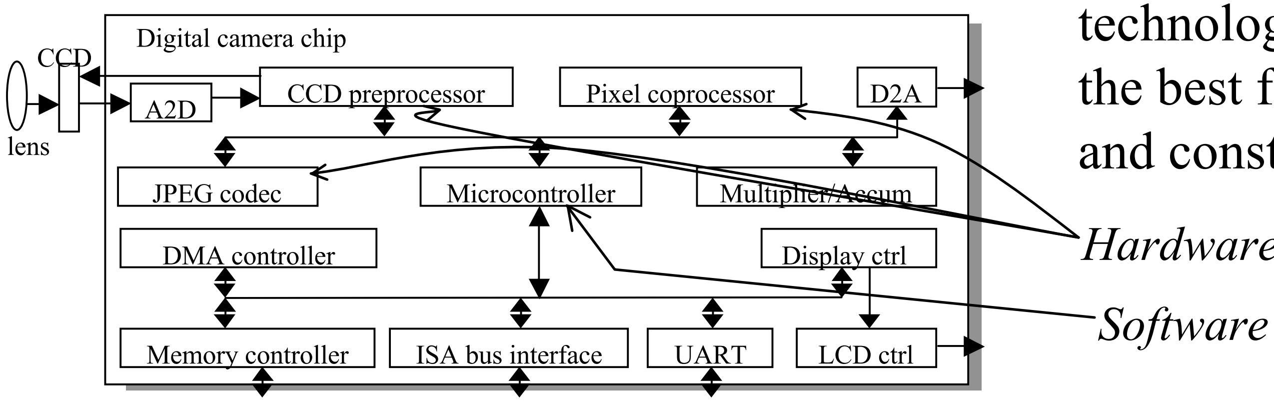
Design challenge – optimizing design metrics

- Common metrics (continued)
  - Time-to-prototype: the time needed to build a working version of the system
  - Time-to-market: the time required to develop a system to the point that it can be released and sold to customers
  - Maintainability: the ability to modify the system after its initial release
  - Correctness, safety, many more

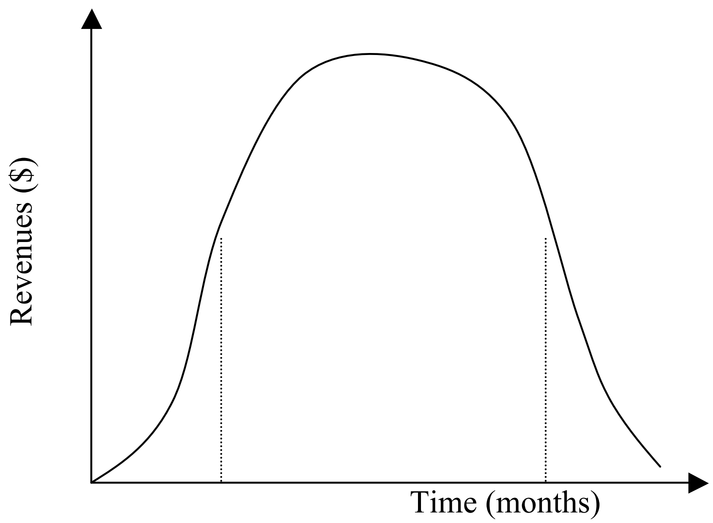
Design metric competition -- improving one may worsen others



- Expertise with both **software and hardware** is needed to optimize design metrics
  - Not just a hardware or software expert, as is common
  - A designer must be comfortable with various technologies in order to choose the best for a given application and constraints

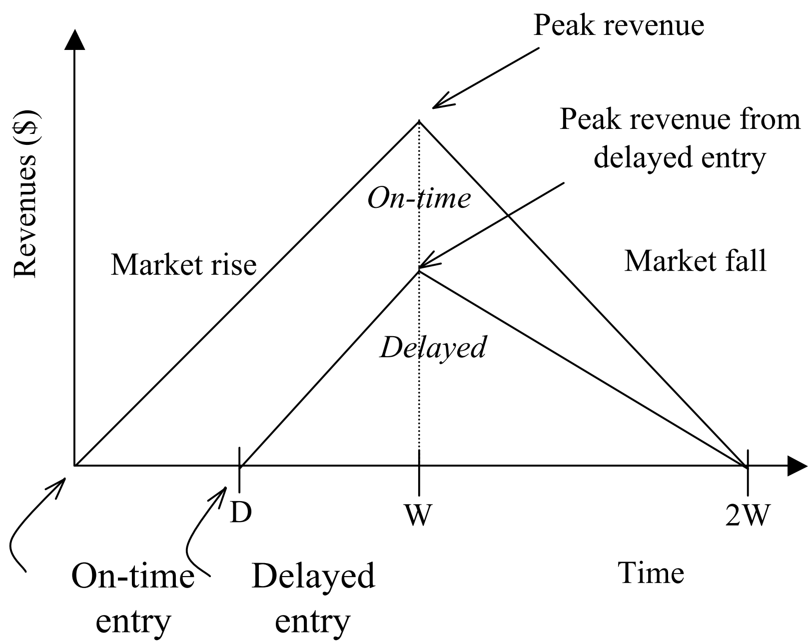


Time-to-market: a demanding design metric



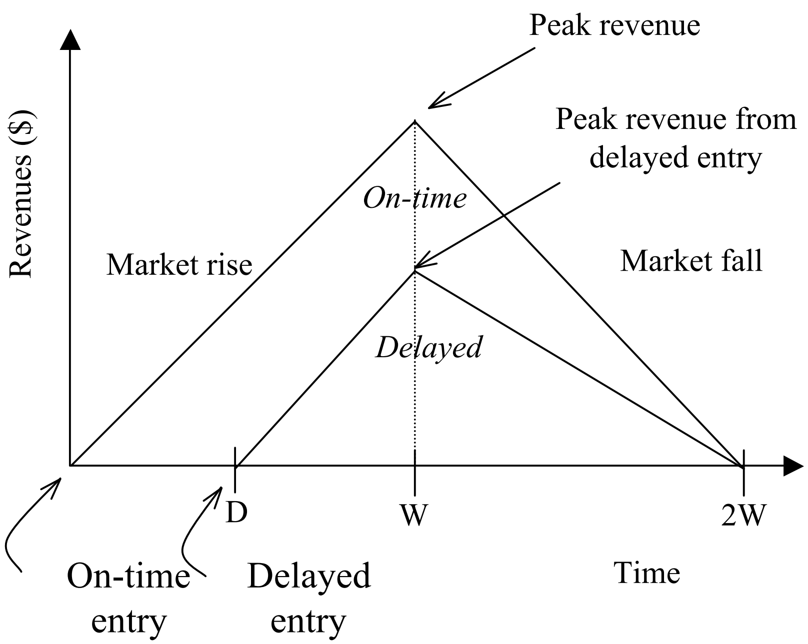
- Time required to develop a product to the point it can be sold to customers
- Market window
  - Period during which the product would have highest sales
- Average time-to-market constraint is about 8 months
- Delays can be costly

Losses due to delayed market entry



- Simplified revenue model
  - Product life = 2W, peak at W
  - Time of market entry defines a triangle, representing market penetration
  - Triangle area equals revenue
- Loss
  - The difference between the on-time and delayed triangle areas

Losses due to delayed market entry (cont.)



- Area = 1/2 * base * height
  - On-time = 1/2 * 2W * W
  - Delayed = 1/2 * (W-D+W)*(W-D)
- Percentage revenue loss = $(D(3W-D)/2W^2)*100\%$
- Try some examples
  - Lifetime 2W=52 wks, delay D=4 wks
  - (4*(3*26 –4)/2*26^2) = 22%
  - Lifetime 2W=52 wks, delay D=10 wks
  - (10*(3*26 –10)/2*26^2) = 50%
  - Delays are costly!

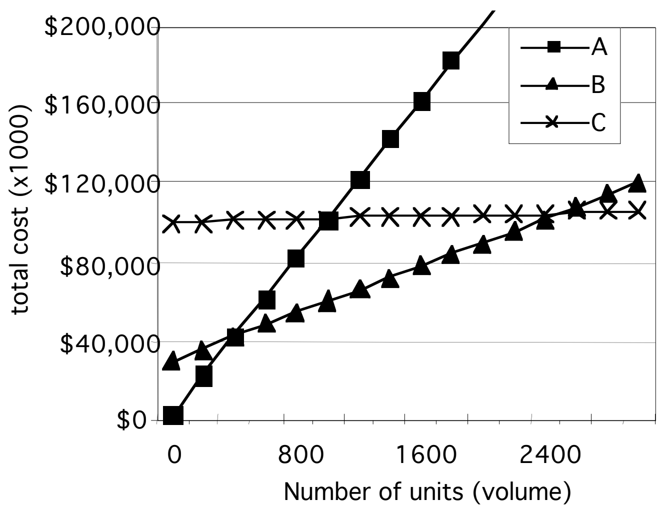
NRE and unit cost metrics

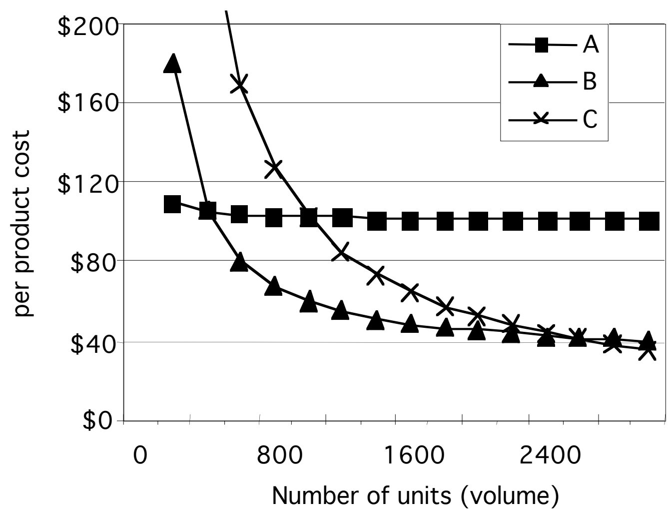
- Costs:
  - Unit cost: the monetary cost of manufacturing each copy of the system, excluding NRE cost
  - NRE cost (Non-Recurring Engineering cost): The one-time monetary cost of designing the system
  - *total cost = NRE cost + unit cost * # of units*
  - *per-product cost = total cost / # of units*
    = (NRE cost / # of units) + unit cost
- Example
  - NRE=$2000, unit=$100
  - For 10 units
    - total cost = $2000 + 10*$100 = $3000
    - per-product cost = $2000/10 + $100 = $300

Amortizing NRE cost over the units results in an additional $200 per unit

NRE and unit cost metrics

- Compare technologies by costs -- best depends on quantity
  - Technology A: NRE=$2,000, unit=$100
  - Technology B: NRE=$30,000, unit=$30
  - Technology C: NRE=$100,000, unit=$2





- But, must also consider time-to-market

The performance design metric

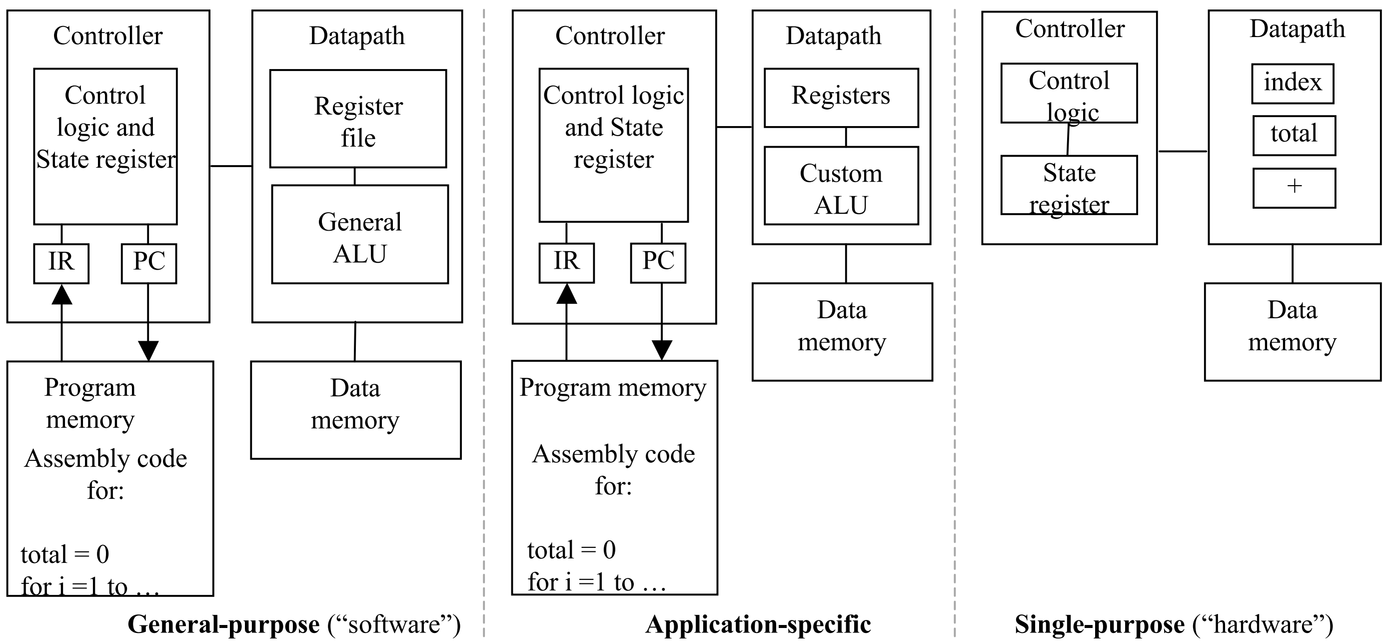
- Widely-used measure of system, widely-abused
  - Clock frequency, instructions per second – not good measures
  - Digital camera example – a user cares about how fast it processes images, not clock speed or instructions per second
- Latency (response time)
  - Time between task start and end
  - e.g., Camera's A and B process images in 0.25 seconds
- Throughput
  - Tasks per second, e.g. Camera A processes 4 images per second
  - Throughput can be more than latency seems to imply due to concurrency, e.g. Camera B may process 8 images per second (by capturing a new image while previous image is being stored).
- *Speedup* of B over S = B's performance / A's performance
  - Throughput speedup = 8/4 = 2

Three key embedded system technologies

- Technology
  - A manner of accomplishing a task, especially using technical processes, methods, or knowledge
- Three key technologies for embedded systems
  - Processor technology
  - IC technology
  - Design technology

Processor technology

- The architecture of the computation engine used to implement a system's desired functionality
- Processor does not have to be programmable
  - "Processor" *not* equal to general-purpose processor

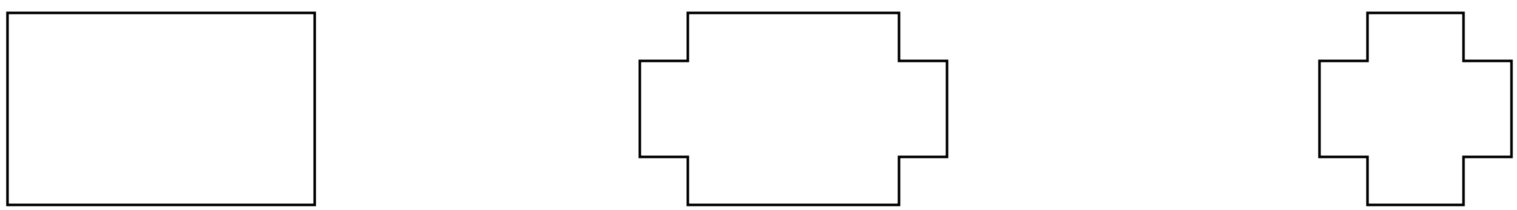


Processor technology

- Processors vary in their customization for the problem at hand

Desired functionality

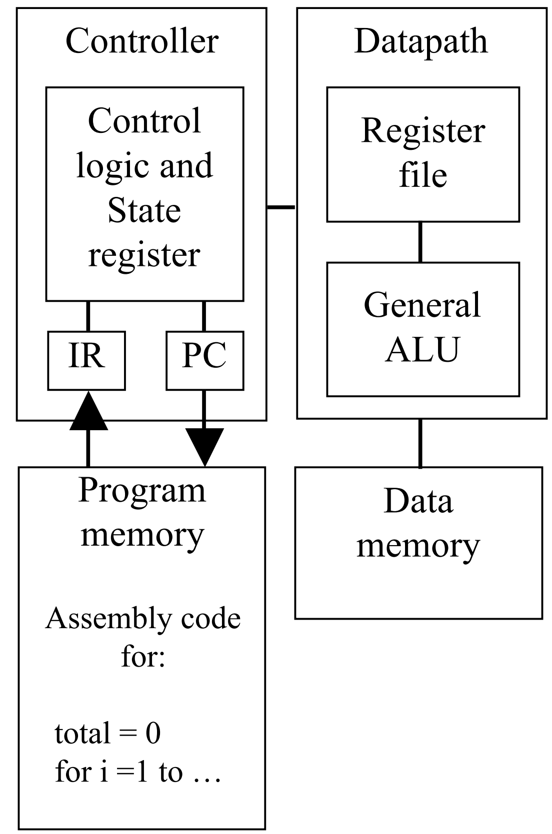
```
total = 0
for i = 1 to N  loop
   total += M[i]
end loop
```



General-purpose processor | Application-specific processor | Single-purpose processor

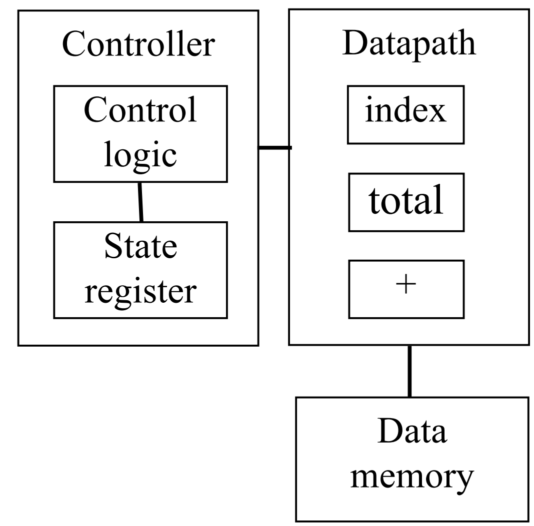
General-purpose processors

- Programmable device used in a variety of applications
  - Also known as "microprocessor"
- Features
  - Program memory
  - General datapath with large register file and general ALU
- User benefits
  - Low time-to-market and NRE costs
  - High flexibility
- "Pentium" the most well-known, but there are hundreds of others



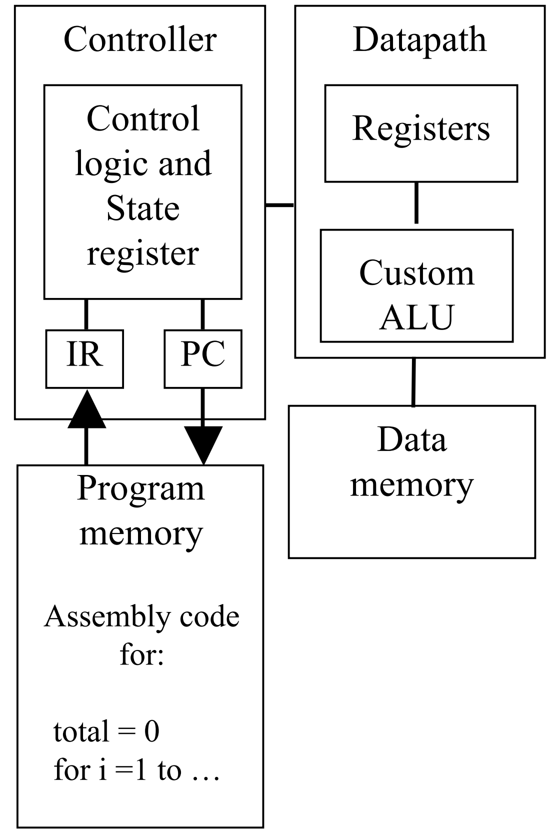
Single-purpose processors

- Digital circuit designed to execute exactly one program
  - a.k.a. coprocessor, accelerator or peripheral
- Features
  - Contains only the components needed to execute a single program
  - No program memory
- Benefits
  - Fast
  - Low power
  - Small size



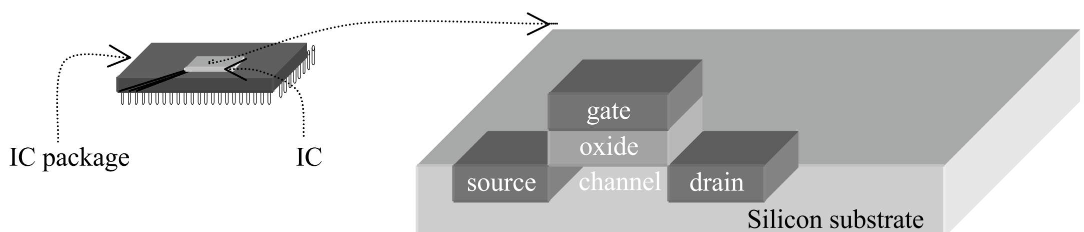
Application-specific processors

- Programmable processor optimized for a particular class of applications having common characteristics
  - Compromise between general-purpose and single-purpose processors
- Features
  - Program memory
  - Optimized datapath
  - Special functional units
- Benefits
  - Some flexibility, good performance, size and power



IC technology

- The manner in which a digital (gate-level) implementation is mapped onto an IC
  - IC: Integrated circuit, or “chip”
  - IC technologies differ in their customization to a design
  - IC’s consist of numerous layers (perhaps 10 or more)
    - IC technologies differ with respect to who builds each layer and when



IC technology

- Three types of IC technologies
  - Full-custom/VLSI
  - Semi-custom ASIC (gate array and standard cell)
  - PLD (Programmable Logic Device)

Full-custom/VLSI

- All layers are optimized for an embedded system's particular digital implementation
  - Placing transistors
  - Sizing transistors
  - Routing wires
- Benefits
  - Excellent performance, small size, low power
- Drawbacks
  - High NRE cost (e.g., $300k), long time-to-market

Semi-custom

- Lower layers are fully or partially built
  - Designers are left with routing of wires and maybe placing some blocks
- Benefits
  - Good performance, good size, less NRE cost than a full-custom implementation (perhaps $10k to $100k)
- Drawbacks
  - Still require weeks to months to develop

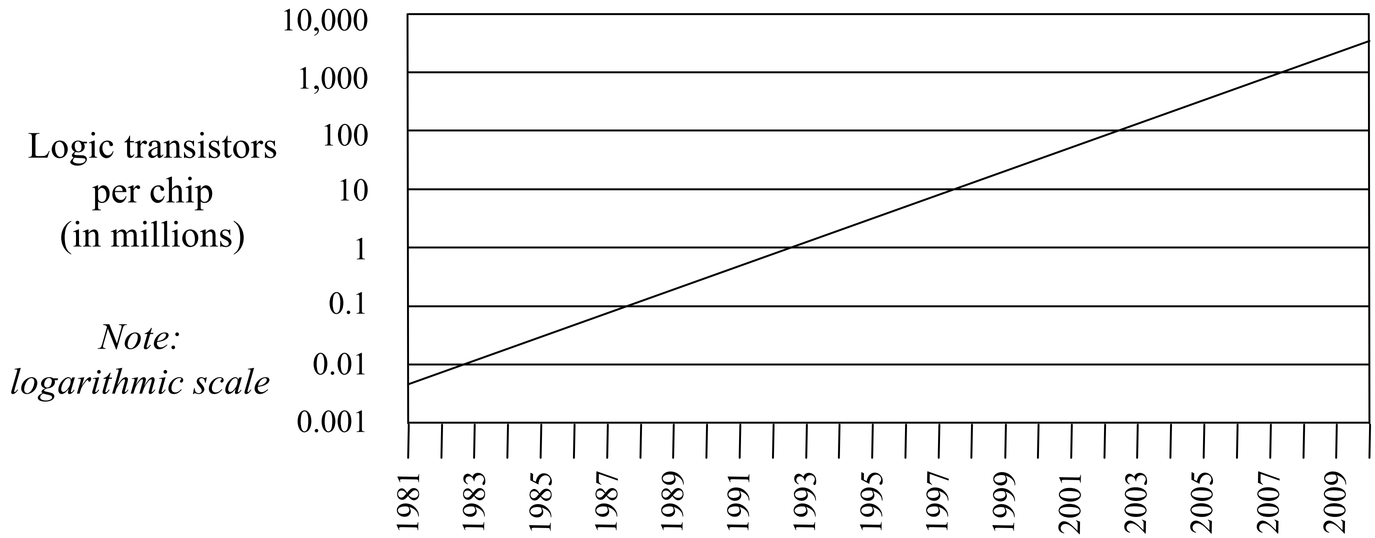
PLD (Programmable Logic Device)

- All layers already exist
  - Designers can purchase an IC
  - Connections on the IC are either created or destroyed to implement desired functionality
  - Field-Programmable Gate Array (FPGA) very popular
- Benefits
  - Low NRE costs, almost instant IC availability
- Drawbacks
  - Bigger, expensive (perhaps $30 per unit), power hungry, slower

Moore's law

- The most important trend in embedded systems
  - Predicted in 1965 by Intel co-founder Gordon Moore

<u>IC transistor capacity has doubled roughly every 18 months for the past several decades</u>

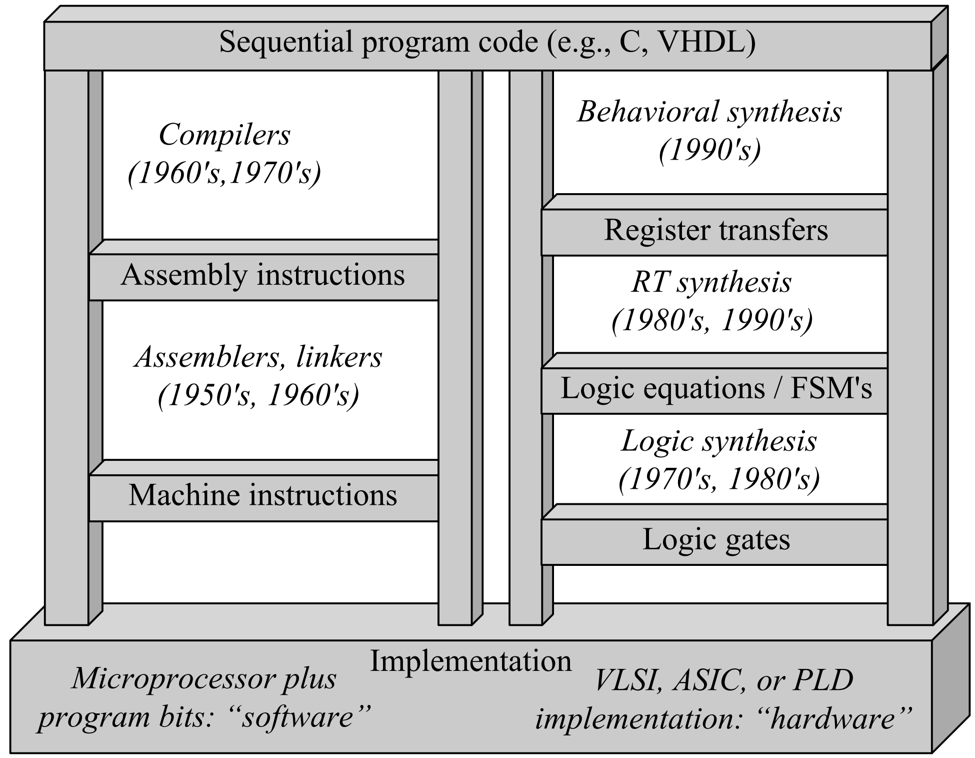


The Future of Moore's law

- Does Moore's Law still hold?
  - This growth rate has been steady until ~2005
  - Factors limiting transistor counts
    - *Power consumption (leakage power)*
    - *Temperature*
    - *Development of new technology (65nm, 45nm, 30nm…)*
- New Trend
  - Multiple computing cores
  - Not significant higher frequency
  - Nanotechnology
- "The law has often met obstacles that appeared insurmountable, before soon surmounting them." - Wikipedia.

The co-design ladder

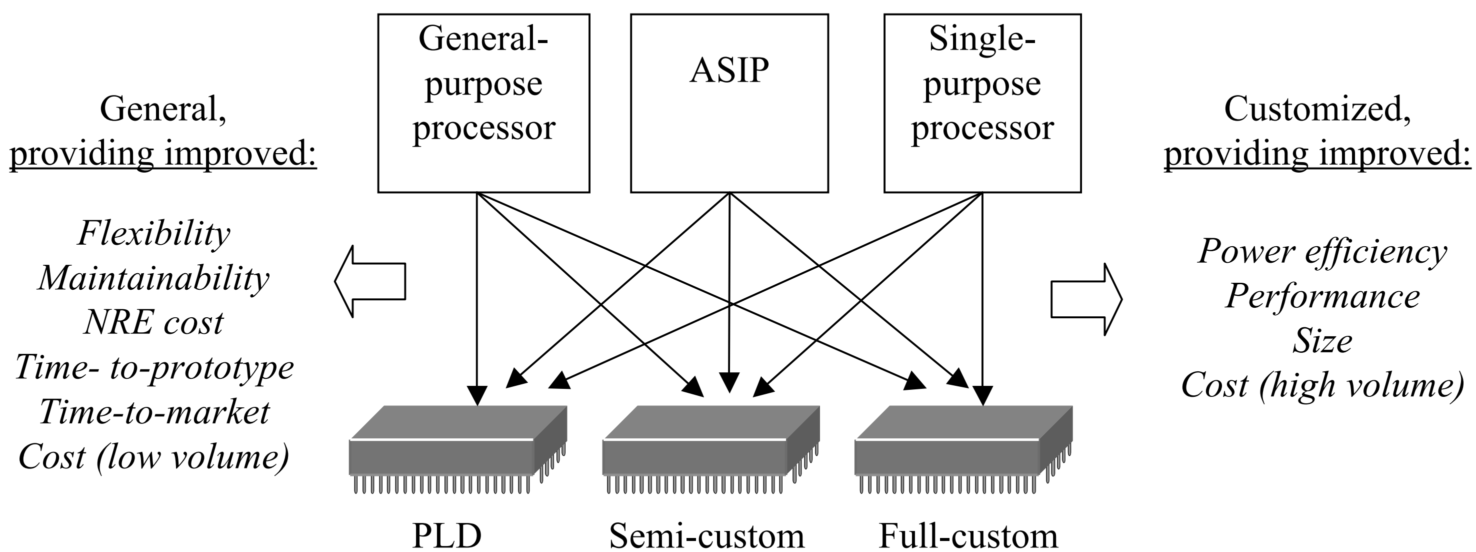
- In the past:
  - Hardware and software design technologies were very different
  - Recent maturation of synthesis enables a unified view of hardware and software
- Hardware/software "codesign"



The choice of hardware versus software for a particular function is simply a tradeoff among various design metrics, like performance, power, size, NRE cost, and especially flexibility; there is no fundamental difference between what hardware or software can implement.

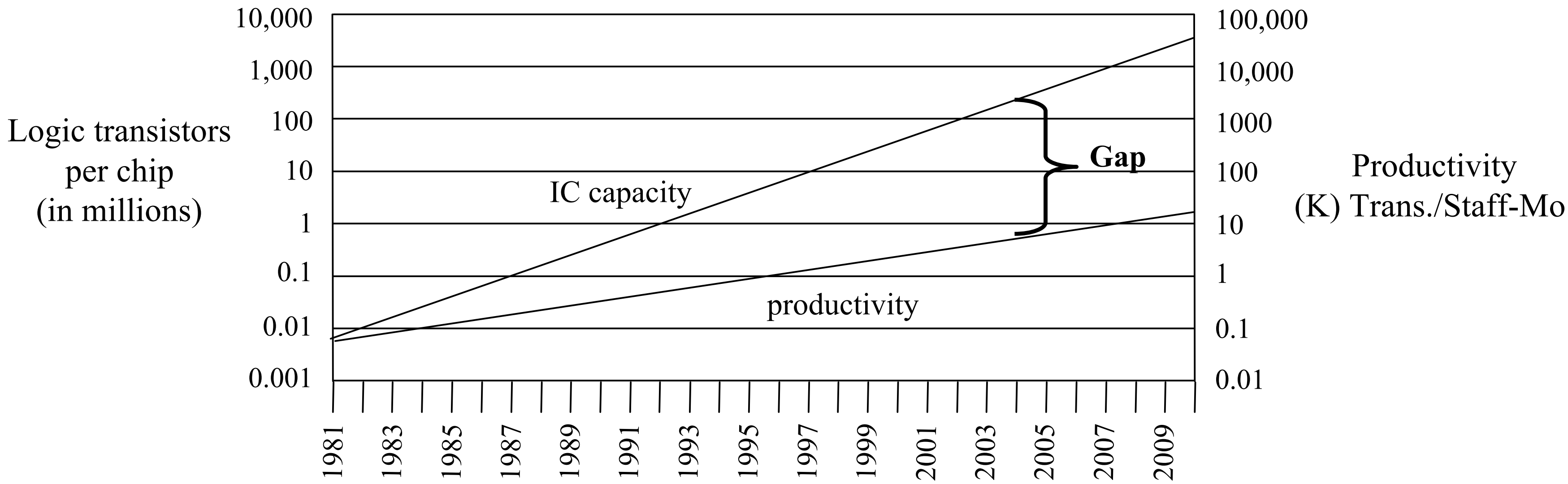
Independence of processor and IC technologies

- Basic tradeoff
  - General vs. custom
  - With respect to processor technology or IC technology
  - The two technologies are independent



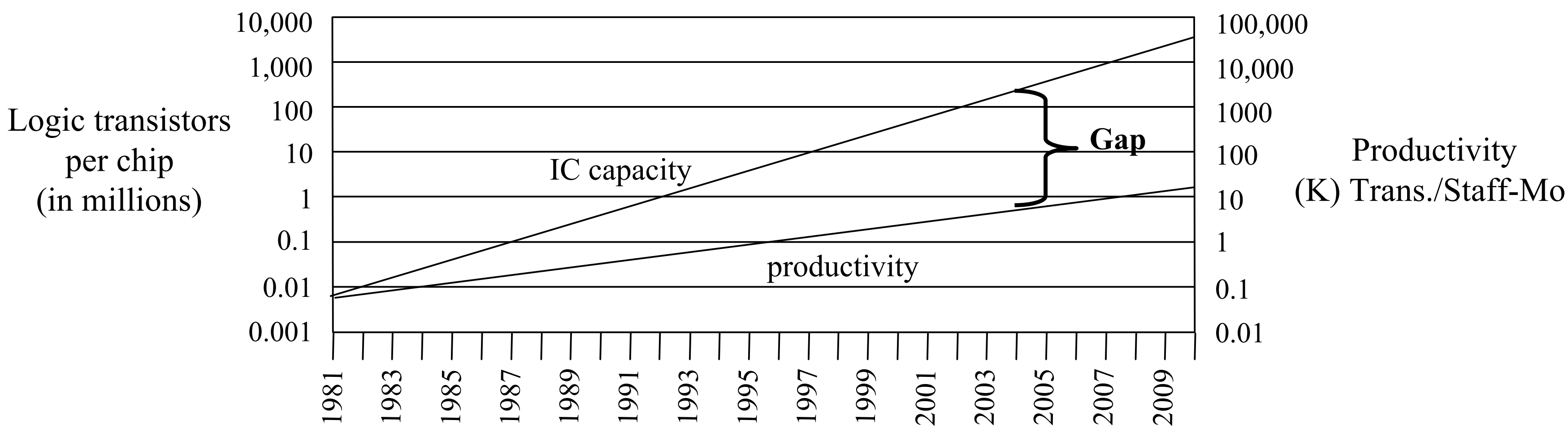
Design productivity gap

- While designer productivity has grown at an impressive rate over the past decades, the rate of improvement has not kept pace with chip capacity



Design productivity gap

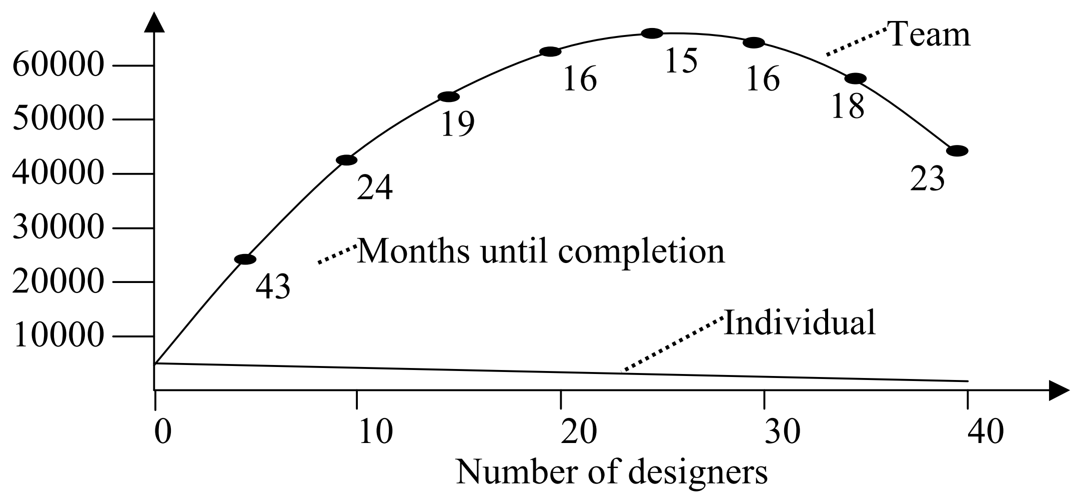
- 1981 leading edge chip required 100 designer months
  - 10,000 transistors / 100 transistors/month
- 2002 leading edge chip requires 30,000 designer months
  - 150,000,000 / 5000 transistors/month
- Designer cost increase from $1M to $300M



The mythical man-month

- The situation is even worse than the productivity gap indicates
- In theory, adding designers to team reduces project completion time
- In reality, productivity per designer decreases due to complexities of team management and communication
- In the software community, known as “the mythical man-month” (Brooks 1975)
- At some point, can actually lengthen project completion time! (“Too many cooks”)

- 1M transistors, 1 designer=5000 trans/month
- Each additional designer reduces for 100 trans/month
- So 2 designers produce 4900 trans/month each



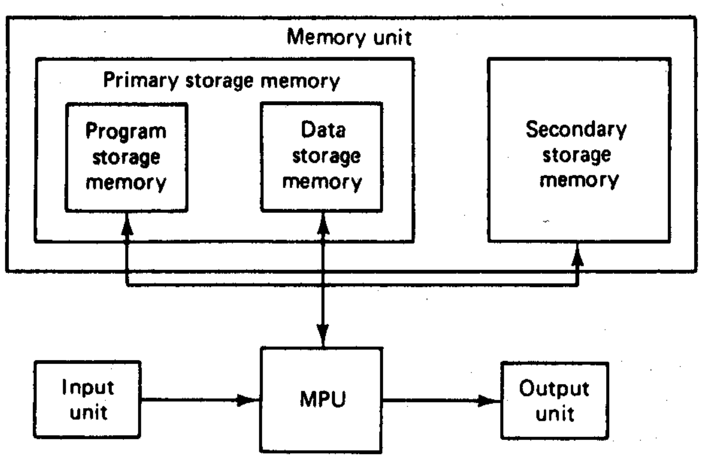
Summary

- Embedded systems are everywhere
- Key challenge: optimization of design metrics
  - Design metrics compete with one another
- A unified view of hardware and software is necessary to improve productivity
- Three key technologies
  - Processor: general-purpose, application-specific, single-purpose
  - IC: Full-custom, semi-custom, PLD, Moore's Law
  - Design: hw/sw co-design, design productivity

16.480/552 Micro II and Embedded Systems Design: Introduction

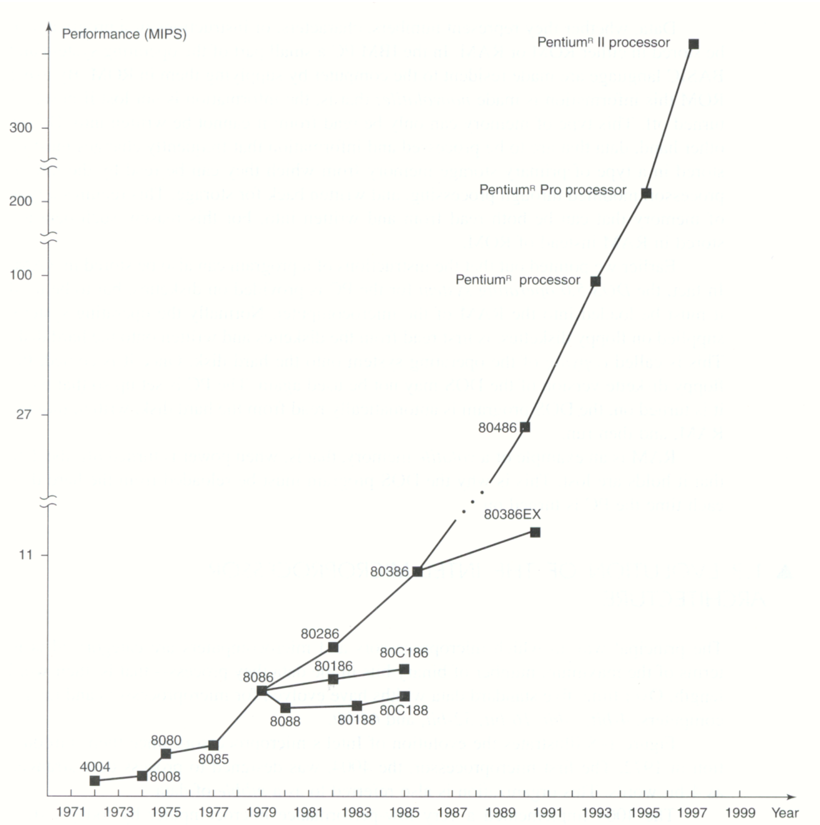
Introduction to Microprocessors and Microcomputers

Elements of a Microcomputer



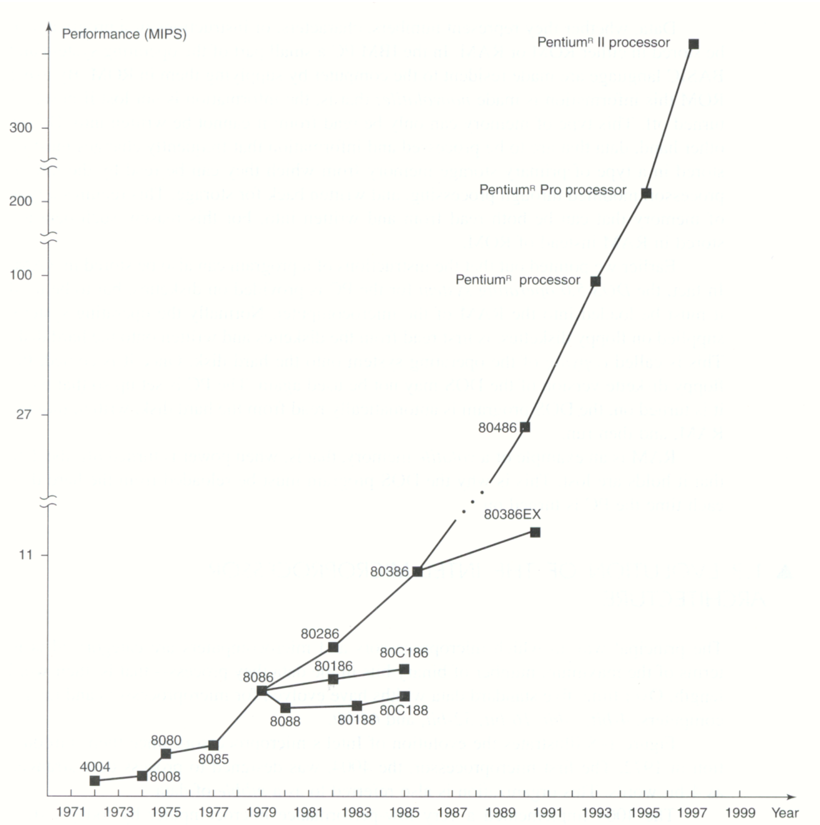
- Hardware of a microcomputer is divided into 4 functions sections
  - Input unit
  - Microprocessing unit
  - Memory unit
  - Output unit
- Microprocessing unit—MPU
  - 8088, ARM, Xscale, 68K, MSP430,
- Memory unit
  - Used to store information such as numbers and characters
  - Memory subsystem is partitioned into
    - Primary storage memory– internal storage
    - Secondary storage memory—external storage

1.3 Evolution of the Intel Microprocessor Architecture– Size and Performance

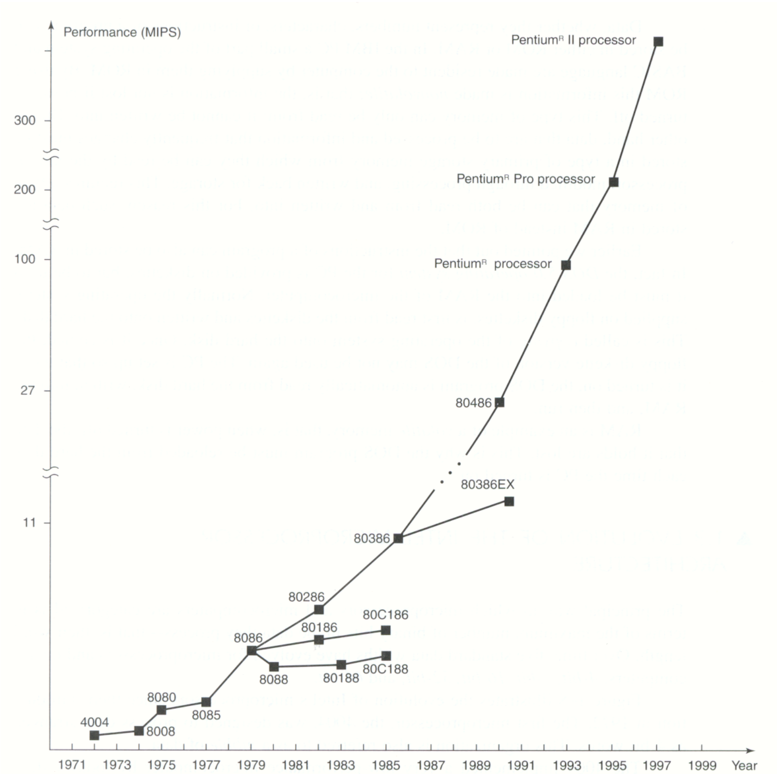


- Standard Sizes of Microprocessors
  - Organized by the maximum size data they can process—4-bit through 64-bit
  - Evolved over time as semiconductor processor technology advanced
- 4-bit Microprocessor: 4004—1972
  - Processes 4-bit data (nibble)
  - Used in calculators
- 8-bit microprocessors: 8080/8085/Z80--1974
  - Processes 8-bit data (byte)
  - Used in instruments, cash registers, and small personal computers
  - Birth of the multi-chip microcomputer—MPU plus special purpose peripheral chips for I/O

Size and Performance

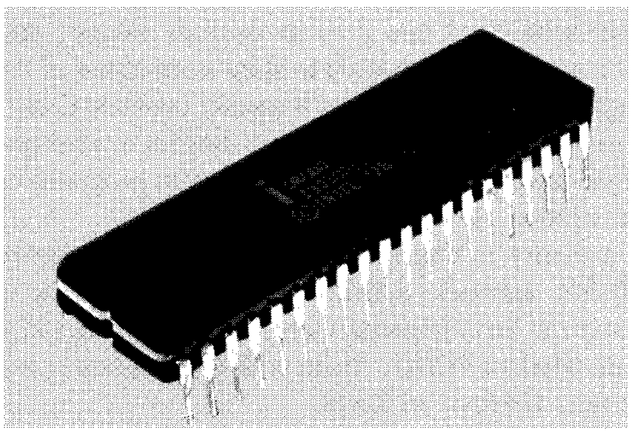


- 16-bit Microprocessor: 8088/8086—1979
  - Processes 16-bit data (word)
  - Wide spread use; birth of the personal computer industry
  - High integration solutions—80188/80186
  - Led to more rapid advances in memory and I/O peripheral technology
- 32-bit microprocessors: 80386, 80486, Pentium—1985-today
  - Processes 32-bit data (double-word)
  - Initiated drive to high performance
  - Advanced architecture taking into account needs of operating systems
- 64-bit microprocessorss: Itanium, Athlon64,



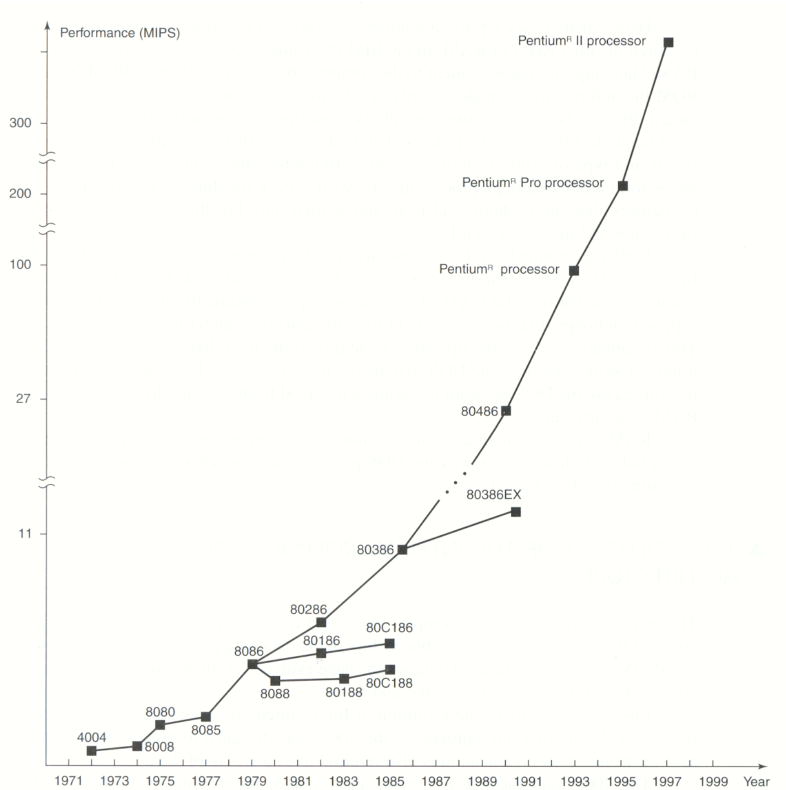
- 64-bit Microprocessor: Itanium—2001
  - Processes 64-bit data (quad word)
  - Applies parallel processing to achieve higher performance at lower clock rate
  - Targeted at multi-processor computer applications—workstations and file servers

1.3 Evolution of the Intel Microprocessor Architecture– 8088/8086 Microprocessors



- First generation 16-bit microprocessor from Intel Corporation
  - 8086 Microprocessor—1979
    - Full 16-bit architecture
    - Internally processed 16-bit data
    - Externally accessed 16-bit wide data memory
  - 8088 Microprocessor—1980
    - Processed 16-bit data internally
    - Accessed 8-bit wide data memory externally
    - Permitted lower cost system solution
    - Resulted in lower performance
    - Selected as the processor for the original IBM PC
  - Replaced by 80286 Microprocessor—1982

Size and Performance



- **Performance did not become a driving issue until late 1980s– 386 to 486 transition**
- **Measure of performance**
  - **MIPS—How many millions of instructions a processor could execute per second**
    - **MIPS at least doubled with each new generation of processor**
      - **80386 ≈ 11MIPS**
      - **80486 ≈ 27 MIPS**
      - **Pentium ≈ 100 MIPS**
    - **Processor performance of a family first improved by internal architectural advances**
    - **Additional improvement achieved by increasing clock rate**
    - **Today the architectural feature driving performance is parallel execution**

Measuring Performance

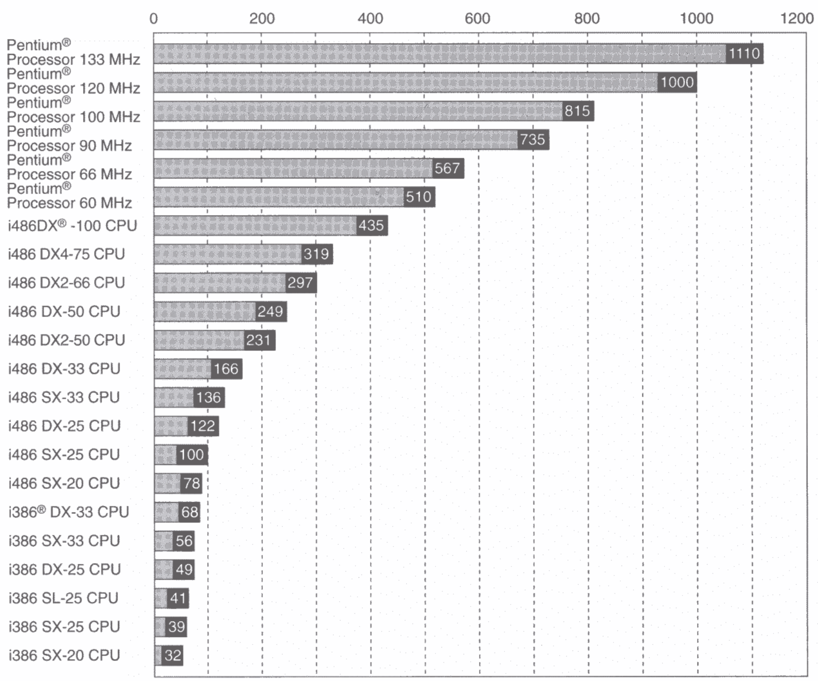
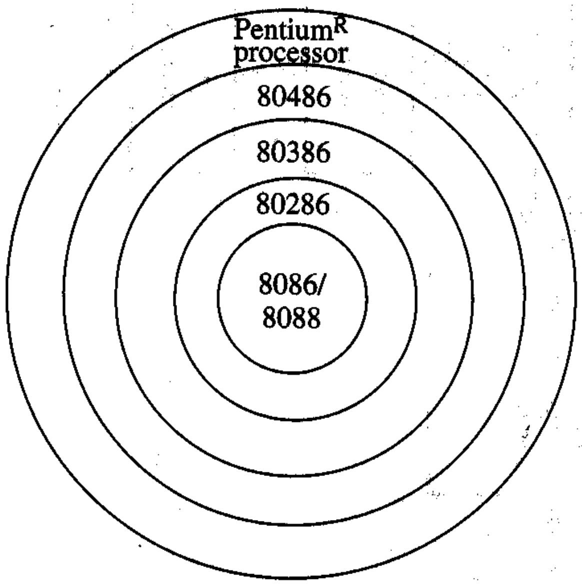


Figure 1–9 iCOMP index rating chart. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1993.).

- Methods
  - MIPS—Processor performance
  - Drystone V1.1– System level performance (compared to performance of VAX 1.1 computer)
  - iCOMP—Intel introduced method of comparing system level performance of 32-bit 80x86 processor based PCs
- iCOMP
  - Rating based on result of a suite of performance components weighted by their normal occurrence in widely used applications
    - **Integer mathematics**
    - **Floating-point mathematics**
    - **Graphics**
    - **Video**
    - **Etc.**
- **SPB Benchmark for Pocket PCs (Homework)**

Software Compatibility

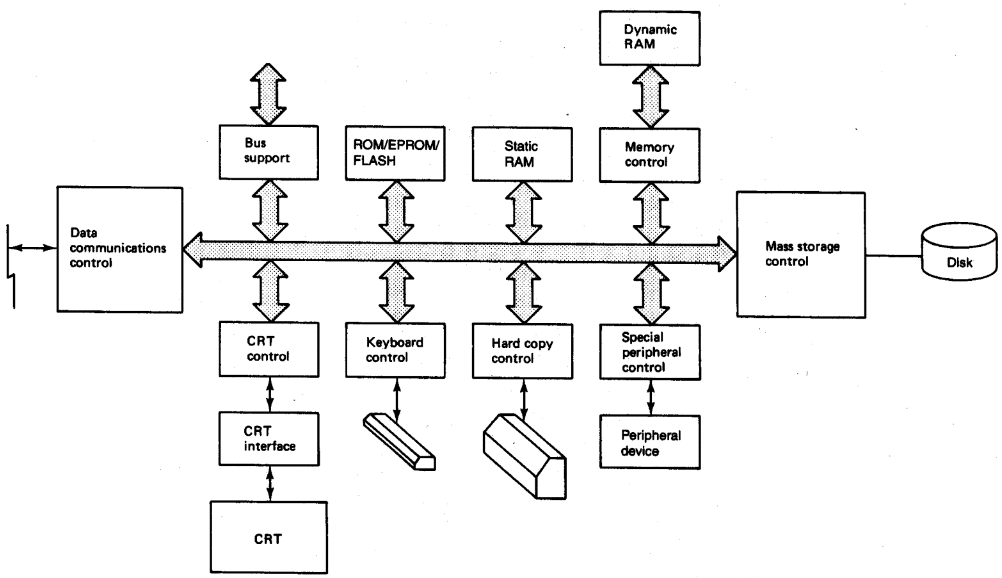


- Compatibility is a critical need of microprocessors designed for reprogrammable applications
  - New family members must be a superset of the earlier
  - Permits application programs and OS written for the earlier members to run unchanged on the new member
- Application Instruction set of 8088/8086 is known as "base instruction set"
  - Significant enhancements in 80286 and 80386
  - Limited changes in 80486 and Pentium
  - Programs written with base instruction set run on all processor families
  - Referred to as "upward compatibility"

Real and Protected Modes

- Real- and protected mode system software architecture introduced with 80286
  - Real-mode– operates like a high-performance 8088/8086
  - Protected mode—implements advanced system architecture for OS
    - System control instruction set introduced with 80286 and enhanced in future generations
    - Much larger memory address space– 1Gbyte for 80286 and 64Tbytes for 80386
    - Memory management
    - Protection
    - Multitasking
    - Virtual 8086 mode

Peripheral Support



- A reprogrammable microcomputer is implemented as a multi-chip microcomputer

  Requires a wide variety of interfaces
  - ROM/RAM memory
  - Keyboard connection
  - Floppy disk drive
  - Hard disk drive
  - Serial communication connections
  - Local area network connection

  Special purpose VLSI devices called "peripherals ICs" have been developed as highly integrated solutions for most of these application

Number Systems— Decimal Number Representation

0
1
2
3
4
5
6
7
8
9

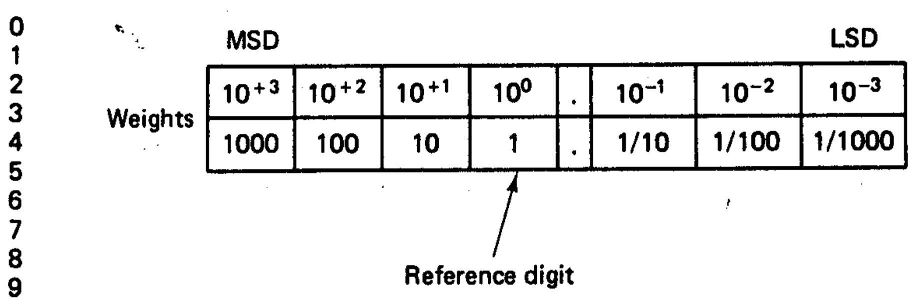
MSD … LSD

Weights	10^{+3}	10^{+2}	10^{+1}	10^{0}	.	10^{-1}	10^{-2}	10^{-3}
	1000	100	10	1	.	1/10	1/100	1/1000

Reference digit

- **Numbers we use everyday are express using the decimal number system**
- **Characteristics of the decimal number system**
  - **Base or radix—number of symbols used—10**
  - **Coefficients—set of symbols used to represent numerical quantities—0, 1,…..9**
  - **Positional notation—digit**
    - **Positional notation that determines the value a symbol represents**
      - **Units digit, Hundreds digit, Tenths digit,……**
  - **Radix point—marks boundary between whole part of number (integer) and fractional part of number—decimal point(.)**
  - **Weight—positional value of a digit—powers of 10**
    - **Units = 10^{0} = 1**
    - **Hundreds = 10^{+2} = 100**
    - **Tenths = 10^{-1} = 1/10 = 0.1**

Decimal Number Representation



- **Example:**

  735.23

  - **Most significant digit (MSD)**
    - **Value = 7**
    - **Digit = Hundreds**
    - **Weight = 10^{+2} = 100**
  - **Least significant digit (LSD)**
    - **Value = 3**
    - **Digit = hundredths**
    - **Weight = 10^{-2} = 1/100 =.01**
  - **Computing the value of a number from its digit values and weights**

$735.23 = 7\times10^{+2}+ 3\times10^{+1}+5\times10^{0}+2\times10^{-1}+3\times10^{-2}$

= 7(100)+3(10)+5(1)+2(.1)+3(.01)

= 700+30+5+.2+.03

735.23 = 735.23

Binary Number Representation

- **Numbers used to describe the operation of microcomputer circuitry and microcomputer assembly language software**
- **Characteristics of the binary number system**
  - **Base (radix)—2**
  - **Coefficients (set of symbols)—0 and 1**
  - **Positional notation—bit (binary digit)**
  - **Radix point—binary point (.)**
  - **Weight (powers of 2)**
    - $2^0 = 1$
    - $2^{+2} = 4$
    - $2^{-1} = 1/2 = 0.5$

MSB LSB

0
1

Weights	2^{+3}	2^{+2}	2^{+1}	2^0	.	2^{-1}	2^{-2}	2^{-3}
	8	4	2	1	.	1/2	1/4	1/8

Reference bit

Range of Binary Integer Numbers

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

- **In microcomputer systems, binary numbers that are used to express address and data have a fixed number of bits**
  - **Address = 20-bits**
  - **Data:**
    - **8-bit = byte**
    - **16-bit = word**
    - **32-bit = double-word**
- **The number of unique binary integers that can be formed with a specific number of bits (n) equals 2^n**
  - **8-bit (byte) = 2^8 = 256**
  - **16-bit (word) = 2^{16} = 65,536 (64K)**
- **Range of possible unsigned integer numbers is given in general as**

  $$2^n-1 \geq \text{Range} \geq 0$$

  - **EX: $2^8-1 \geq$ 8-bit (byte) ≥ 0**

    $(256-1) \geq$ 8-bit (byte) ≥ 0

    $255 \geq$ 8-bit (byte) ≥ 0

Decimal Equivalent of a Binary Number

Decimal number	Binary number
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

- **All values may be expressed as both decimal and binary numbers**
- **Number system of a value is identified by a subscript equal to the weight placed to the left of the LSB (if not show base-10 understood)**
  - **Examples:**

    $0_{10} = 0_2$

    $1_{10} = 1_2$

    $3_{10} = 11_2$

    $10_{10} = 1010_2$
- **In the study of microcomputer circuits and software operation, it is frequently necessary to convert numbers between binary and decimal forms.**

Decimal Equivalent of a Binary Number

- **Finding the decimal equivalent of a binary number**
  - **Multiply the value in each bit by its weight**
  - **Add the results of the individual products for each bit**

$$1100_2 = 1(2^{+3}) + 1(2^{+2}) + 0(2^{+1}) + 0(2^0)$$
$$= 1(8) + 1(4) + 0(2) + 0(1)$$
$$= 8 + 4$$
$$1100_2 = 12_{10}$$

Example 1: $1100_2 = ?_{10}$

		product
ost significant bit	$1 \times 2^{+3} = 1(8) =$	**8**
	$1 \times 2^{+2} = 1(4) =$	**4**
	$0 \times 2^{+1} = 0(2) =$	**0**
Least significant bit	$0 \times 2^{0} = 0(1) =$	**0**
		sum 12

$1100_2 = 12_{10}$

Decimal Equivalent of a Binary Number

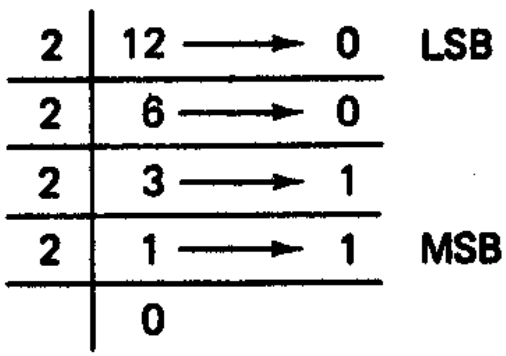
$$
\begin{aligned}
101.01 &= 1(2^{+2}) + 0(2^{+1}) + 1(2^{0}) + 0(2^{-1}) + 1(2^{-2}) \\
&= 1(4) + 0(2) + 1(1) + 0(\tfrac{1}{2}) + 1(\tfrac{1}{4}) \\
&= 4 + 1 + .25 \\
101.01 &= 5.25_{10}
\end{aligned}
$$

- **Example 2: $101.01_2 = ?_{10}$**

		product
Most significant bit	$1\times2^{+2} = 1(4)$	**= 4**
	$0\times2^{+1} = 0(2)$	**= 0**
	$1\times2^{0} = 1(1)$	**= 1**
	$0\times2^{-1} = 0(1/2)$	**= 0**
Least significant bit	$1\times2^{-2} = 1(1/4)$	**= 0.25**
	sum	**5.25**

$101.01_2 = 5.25_{10}$

Binary Equivalent of an Integer Decimal Number



$12_{10} = 1100_2$

Finding the binary equivalent of an integer decimal number—successive division process

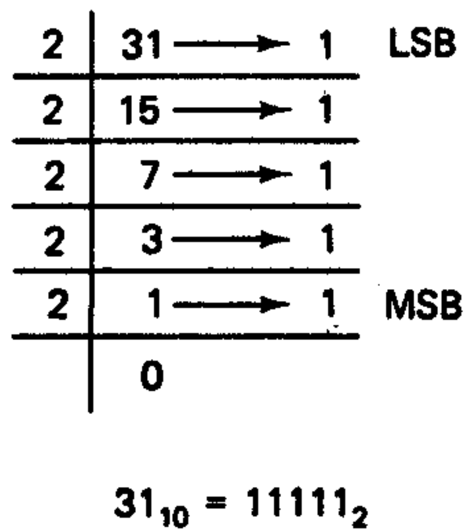
1. **Divide the decimal number by 2; bring out the remainder to the right as the coefficient for the corresponding binary bit**
2. **Repeat this division operation on each result until the quotient is 0; each time using the remainder as the coefficient of the binary bit**
3. **Collect the coefficients to form the binary number**
   - **First remainder is the least significant bit**
   - **Last remainder is the most significant bit**

Example 1: $12_{10} = ?_2$

	quotient	remainder
12 ÷ 2 = 6		**→ 0 LSB**
6 ÷ 2 = 3		**→ 0**
3 ÷ 2 = 1		**→ 1**
1 ÷ 2 = 0		**→ 1 MSB**

$12_{10} = 1100_2$

Binary Equivalent of an Integer Decimal Number



- **Example 2: 31_{10} = ?**

quotient	remainder
31 ÷ 2 = 15	**→ 1 LSB**
15 ÷ 2 = 7	**→ 1**
7 ÷ 2 = 3	**→ 1**
3 ÷ 2 = 1	**→ 1**
1 ÷ 2 = 0	**→ 1 MSB**

$31_{10} = 11111_2$

Hexadecimal Number Representation

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

MSD LSD

Weights	16^{+3}	16^{+2}	16^{+1}	16^{0}	.	16^{-1}	16^{-2}
	4096	256	16	1	.	1/16	1/256

Reference digit

- **Inputs and outputs of logic and microcomputer circuitry that involve many bits are frequently expressed as hexadecimal numbers instead of binary numbers for compactness**
  - **Addresses**
  - **Data**
  - **Instruction code**
- **Characteristics of the hexadecimal number system**
  - **Base (radix)—16**
  - **Coefficients (set of symbols)—0 through 9 and A through F**
  - **Positional notation—hexadecimal digit**
  - **Radix point—hexadecimal point (.)**
  - **Weight (powers of 16)**
    - $16^{0} = 1$
    - $16^{+2} = 256$
    - $16^{-1} = 1/16 = 0.0625$

Equivalent Decimal, Binary and Hexadecimal Numbers

Decimal number	Binary number	Hexadecimal number
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

- **All values may be expressed as decimal, binary, and hexadecimal numbers**
- **Binary equivalent of hexadecimal numbers are expressed 4-bits wide**
- **Examples:**

  $1_{16} = 0001_2$

  $7_{16} = 0111_2$

  $9_{16} = 1001_2$

  $A_{16} = 1010_2$

  $F_{16} = 1111_2$

Equivalent Decimal, Binary and Hexadecimal Numbers

Binary number	Hexadecimal number
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

MSB LSB

$2^{11}\ 2^{10}\ 2^9\ 2^8$	$2^7\ 2^6\ 2^5\ 2^4$	$2^3\ 2^2\ 2^1\ 2^0$	Bits
16^2	16^1	16^0	Digits

MSD LSD

1001 : 0000 : 1110
9 : 0 : E

$100100001110_2 = 90E_{16}$

- **Hexadecimal equivalent of a binary number—grouping of bits**
  1. **Starting from the binary point, separate bits into groups of four**
  2. **Replace each group of four binary bits by its equivalent hexadecimal number**
  3. **Most significant 0s may be ignored**
- **Example 1:**

  $100100001110_2 = ?_{16}$

Solution:

Equivalent Decimal, Binary and Hexadecimal Numbers

- **Example 2:**
  $0000011011110001_2 = ?_{16}$

Solution:

0000	:	0110	:	1111	:	0001
0	:	6	:	F	:	1

$0000011011110001_2 = 6F1_{16}$

Equivalent Decimal, Binary and Hexadecimal Numbers

Binary number	Hexadecimal number
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

MSB LSB

2^{11} 2^{10} 2^9 2^8	2^7 2^6 2^5 2^4	2^3 2^2 2^1 2^0	Bits
16^2	16^1	16^0	Digits

MSD LSD

- **Binary equivalent of a hexadecimal number**
  1. **Replace the value in each hexadecimal digit with its 4-bit binary equivalent**
  2. **Most significant 0s may be ignored**

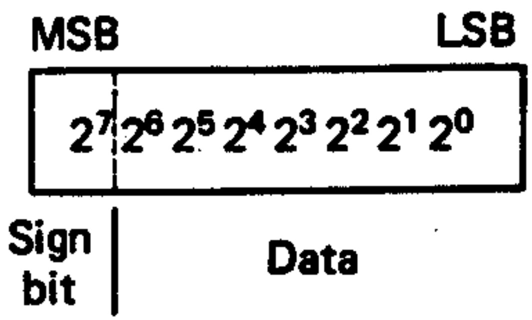
Example 1: $A5_{16} = ?_2$

A : 5

1010 : 0101

$A5_{16} = 10100101_2$

Signed Integer Numbers



- **Data processed by a microcomputer can also represent signed integer numbers**
- **Signed-magnitude format of a signed integer**
  - **The most significant bit of the element of data is called the sign bit**
    - **Sign bit = 0 → + integer number**
    - **Sign bit = 1 → – integer number**
  - **Byte-wide signed integer**
    - **MSB = sign**
    - **7 less significant bits = magnitude of the number**
    - **Range of value for a byte-wide signed integer**

$$+(2^{n-1}-1) \geq \text{Range} \geq -(2^{n-1}-1)$$

Ex. N=8

$$+(2^{7}-1) \geq \text{Range} \geq -(2^{7}-1)$$

$$+127 \geq \text{Range} \geq -127$$

  - **Word-wide signed integer—MSB =Sign and 15-bit magnitude**
  - **Double word-wide signed integer—MSB =Sign and 31-bit magnitude**

Signed Integer Numbers

- **Example:**

Find the value of the byte-wide signed number

01110000_2

Solution:

$$0 : 1110000$$

$$+ : 1(2^6) + 1(2^5) + 1(2^4) + 0(2^3) + 0(2^2) + 0(2^1) + 0(2^0)$$

$$+ : 64 + 32 + 16$$

$$01110000 = +112$$

Binary Encoded Decimal Code

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

(a)

Bits	A_3	A_2	A_1	A_0
Weights	2^3	2^2	2^1	2^0
	8	4	2	1

(b)

- **Binary coded decimal (BCD) code**
  - **BCD coded numbers directly added and subtracted by microcomputer**
  - **Unique 4-bit code $A_3A_2A_1A_0$ represents the ten decimal numbers—0 through 9**
    - **0 through 9 are simply coded with their equivalent 4-bit binary values**
      - **$0_{10} = 0000_2$**
      - **$1_{10} = 0001_2$**
      - **$9_{10} = 1001_2$**
    - **Called "weighted code"**
      - **Each bit is assigned a "weight"—2^0 through 2^3**
      - **Each binary combination is numerically equivalent to the decimal number it represents**

Binary Encoded Decimal Code

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

(a)

Bits	A_3	A_2	A_1	A_0
Weights	2^3	2^2	2^1	2^0
	8	4	2	1

(b)

- **Example 1:**

  Show how the decimal number 84 is coded as a byte-wide BCD coded number.

  Solution: $8 = 1000_{BCD}$

  $4 = 0100_{BCD}$

  $84_{10} = 10000100_{BCD}$

- **Example 2:**

  Find the decimal value of the BCD coded number

  00010010_{BCD}

  Solution:

  $0001_{BCD}\ 0010_{BCD} = 12_{10}$

ASCII Code

				b_7	0	0	0	0	1	1	1	1
				b_6	0	0	1	1	0	0	1	1
				b_5	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1	H_1 / H_0	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	–	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	Λ	n	~
1	1	1	1	F	SI	US	/	?	O	–	o	DEL

- **Alphanumeric character codes—a code used to express a complete set of alphanumeric characters**
  - **Numbers 0-9**
  - **Lower case (a-z) and upper case (A- Z) letters**
  - **Special symbols (+,–, @,etc.)**
  - **Control characters (DEL=delete, BS=backspace, ESC=escape, etc.)**
  - **As many as 128 unique symbols**
- **Alphanumeric character code—requires 7-bit code**

  $2^K \geq 128$

  $2^7 = 128$

  $k = 7$

  - **Gives 128 unique combinations**

1.4 Alphanumeric Codes—ASCII Code

				b_7	0	0	0	0	1	1	1	1
				b_6	0	0	1	1	0	0	1	1
				b_5	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1	H_1 / H_0	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	−	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	∧	n	~
1	1	1	1	F	SI	US	/	?	O	—	o	DEL

- American Standard Code for Information Interchange (ASCII)
  - **Most widely used alphanumeric code**
    - **Databases are mostly character data and normally coded in ASCII**
      - **Person's name**
      - **Address**
      - **Phone number**
      - **Email address**
    - **Widely used for coding of information for transfer over a communication line**
  - **Some microprocessors can process data directly in ASCII coded form**

ASCII Code

				b_7	0	0	0	0	1	1	1	1
				b_6	0	0	1	1	0	0	1	1
				b_5	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1	H_1 / H_0	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	—	o	DEL

- **Expressing character information in ASCII**
  - **Seven bits of an ASCII character are denoted**

    $b_7b_6b_5b_4b_3b_2b_1$
  - **Finding the ASCII code for a character**
    - **Locate the character in the table**
    - **Determine the three MSBs ($b_7b_6b_5$) from the code for the column at the top edge**
    - **Determine the four LSBs ($b_4b_3b_2b_1$) from the code for that row at the left edge**
    - **Combine the column and row codes to form the 7-bit ASCII code**
- **Example 1: How is CR coded in ASCII? Express as a hexadecimal number.**

  $b_7b_6b_5 = 000 \rightarrow$ **Column**

  $b_4b_3b_2b_1 = 1101 \rightarrow$ **Row**

  $b_7b_6b_5b_4b_3b_2b_1 = 0001101 = 0D_{16}$

ASCII Code

				b_7	0	0	0	0	1	1	1	1
				b_6	0	0	1	1	0	0	1	1
				b_5	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1	H_1 / H_0	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	'	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l		
1	1	0	1	D	CR	GS	−	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	−	o	DEL

- **Example:**

How is the computer statement that follows coded in ASCII?

LET Y=X+1

Solution:

$L = 1001100_2 = 4C_{16}$

$E = 1000101_2 = 45_{16}$

$T = 1010100_2 = 54_{16}$

$SP = 0100000_2 = 20_{16}$

$Y = 1011001_2 = 59_{16}$

$= \ = 0111101_2 = 3D_{16}$

$X = 1011000_2 = 58_{16}$

$+ \ = 0101011_2 = 2B_{16}$

$1 = 0110001_2 = 31_{16}$