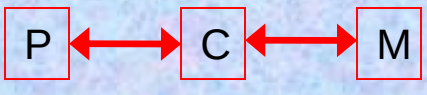
Fundamentals needed before designing supercomputers and other parallel processing systems

Joseph T Wunderlich PhD

Computer Architectures

P = Processor (CPU)
C = Cache
M = Main Memory (RAM)

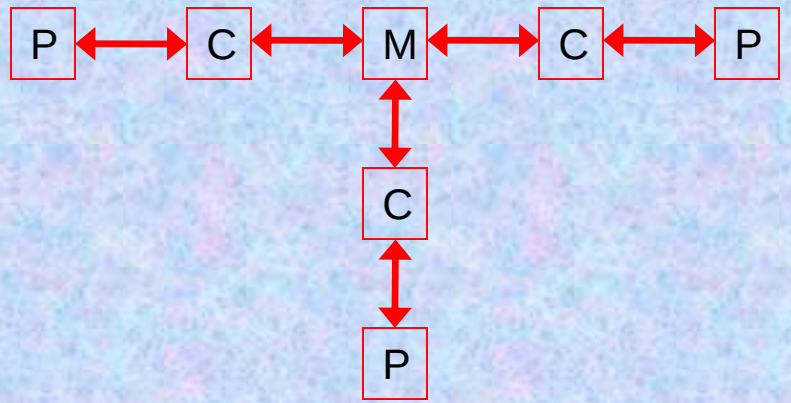
Simple Single Processor



Computer Architectures

P = Processor (CPU)
C = Cache
M = Main Memory (RAM)

- SMP (**S**ymmetric **M**ulti-**P**rocessing)

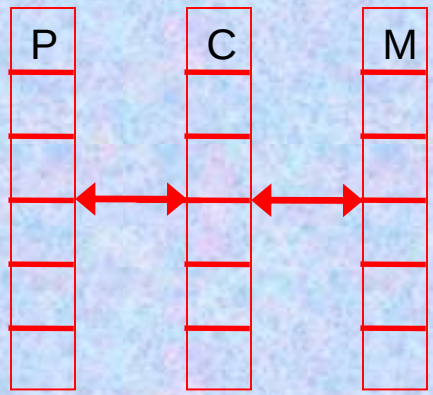


Computer Architectures

P = Processor (CPU)
C = Cache
M = Main Memory (RAM)

Vector Register

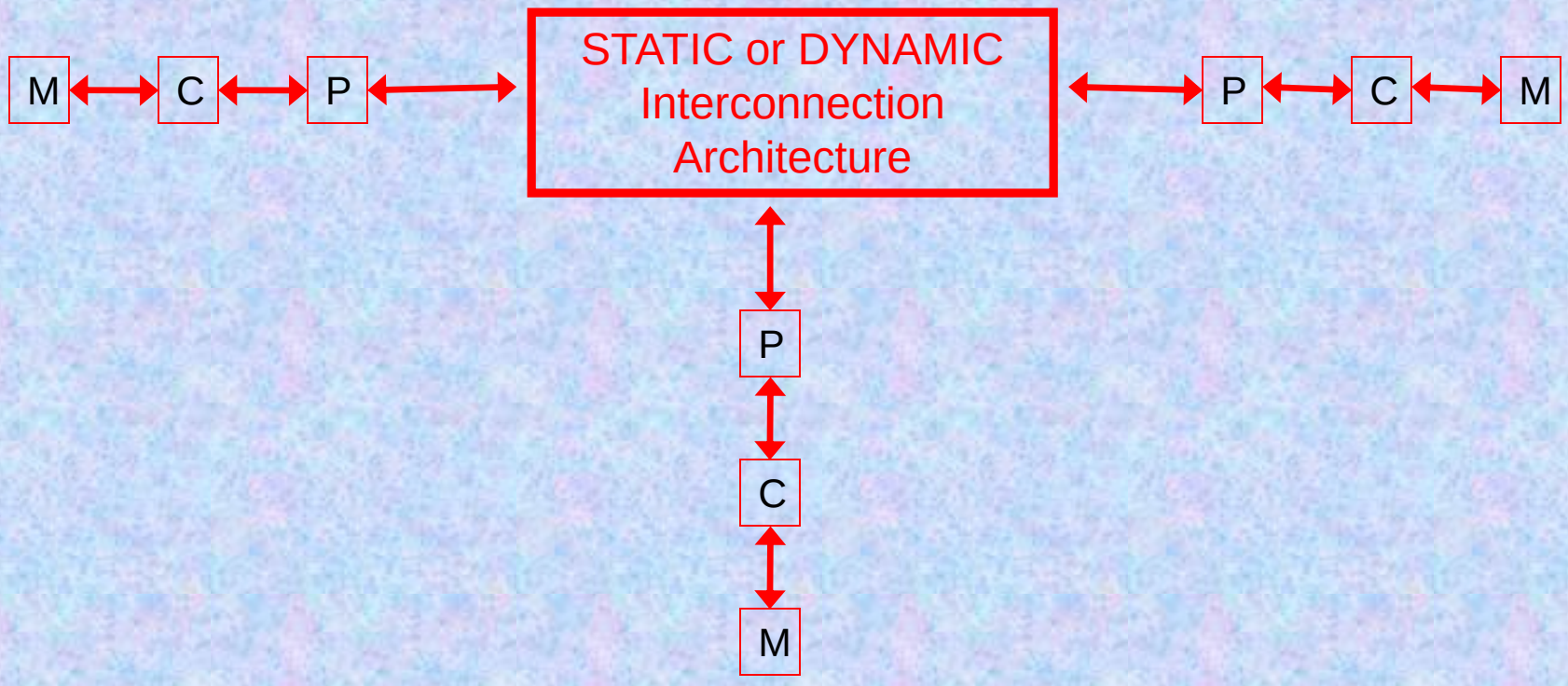
- Multiple functional units in Processor for arithmetic and logic
- Multiple data elements in Cache and main Memory



Computer Architectures

P = Processor (CPU)
C = Cache
M = Main Memory (RAM)

MPP (**M**assively **P**arallel **P**rocessing)



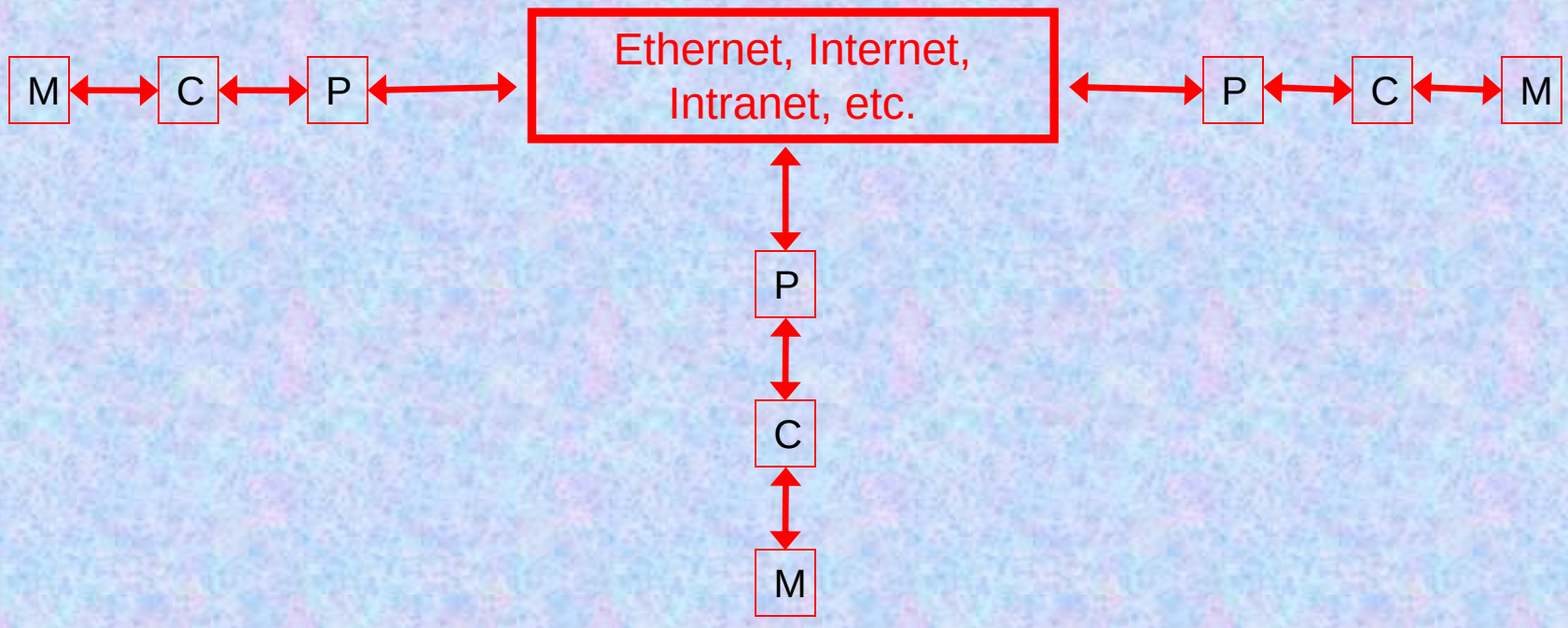
Computer Architectures

P = Processor (CPU)

C = Cache

M = Main Memory (RAM)

Large network dedicated to a single task



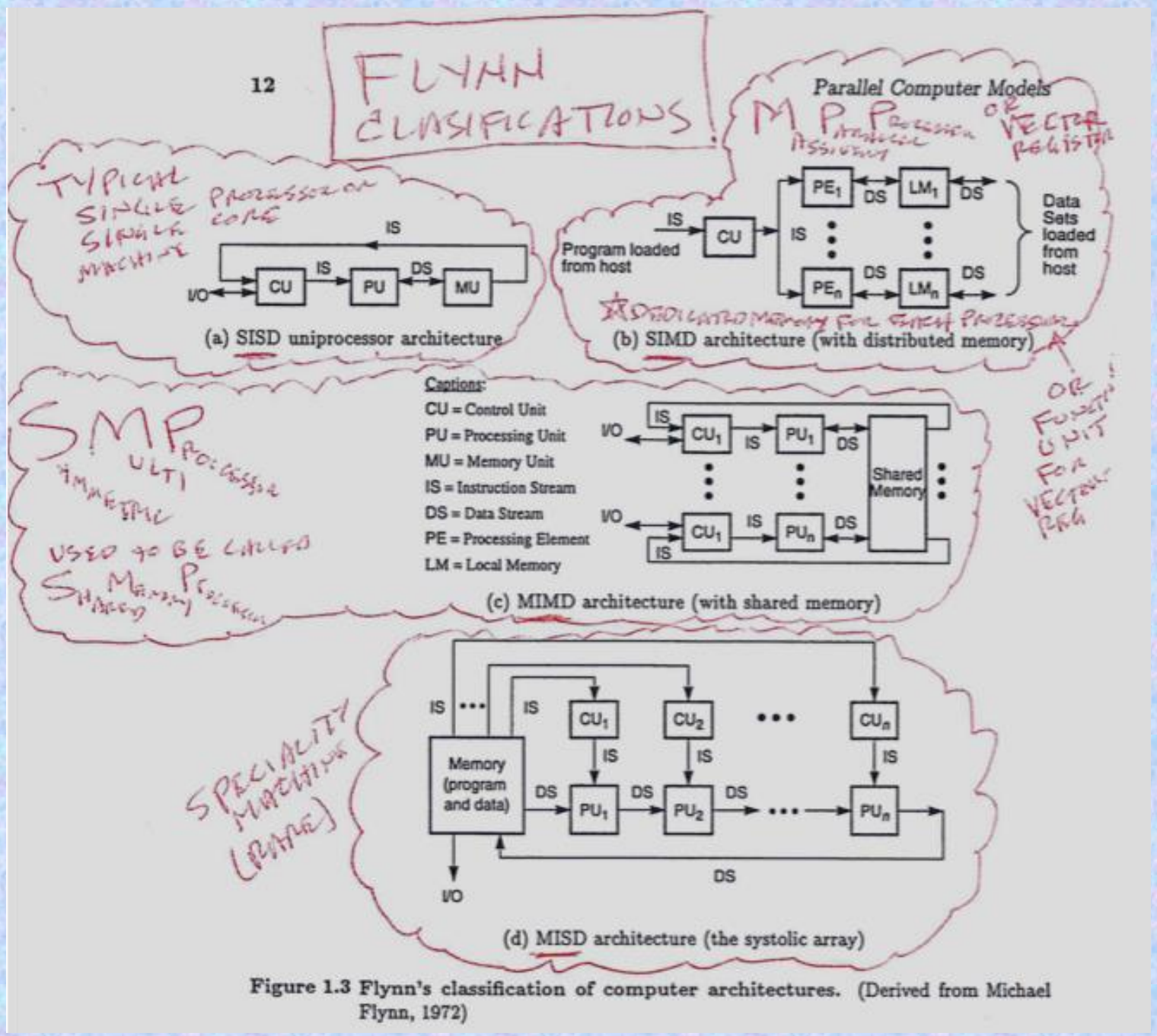


Figure 1.3 Flynn's classification of computer architectures. (Derived from Michael Flynn, 1972)

K. Hwang, ***" Advanced Computer Architecture: Parallelism, Scalability, Programmability,"*** Boston, MA: McGraw-Hill, 1992.

INTERCONNECT ARCHITECTURES of Supercomputers & Parallel Systems

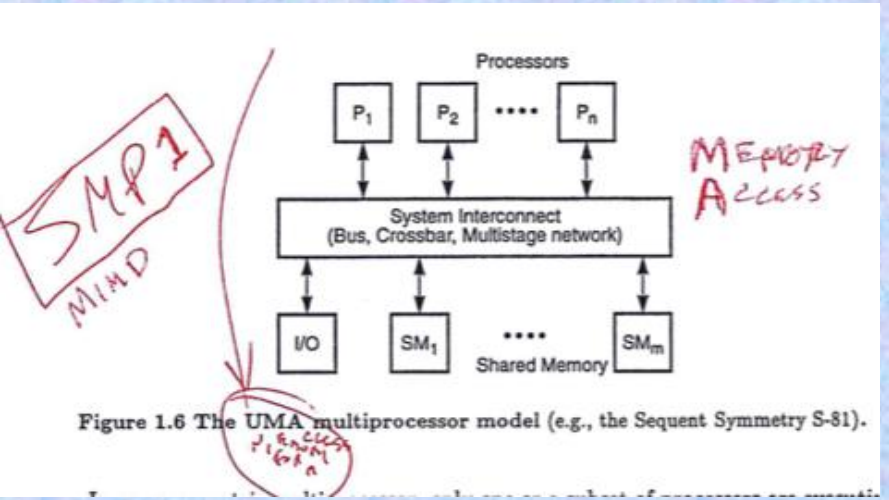
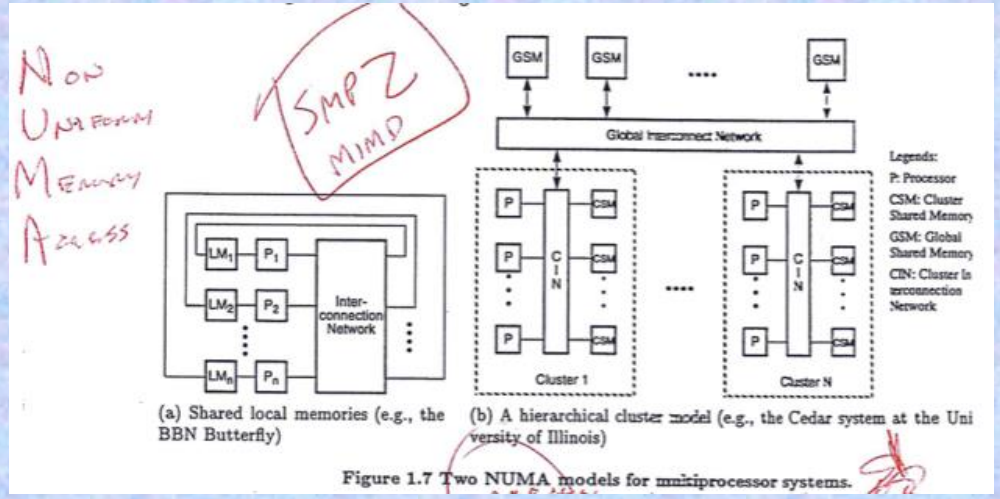


Figure 1.6 The UMA multiprocessor model (e.g., the Sequent Symmetry S-81).



(a) Shared local memories (e.g., the BBN Butterfly)

(b) A hierarchical cluster model (e.g., the Cedar system at the University of Illinois)

Figure 1.7 Two NUMA models for multiprocessor systems.

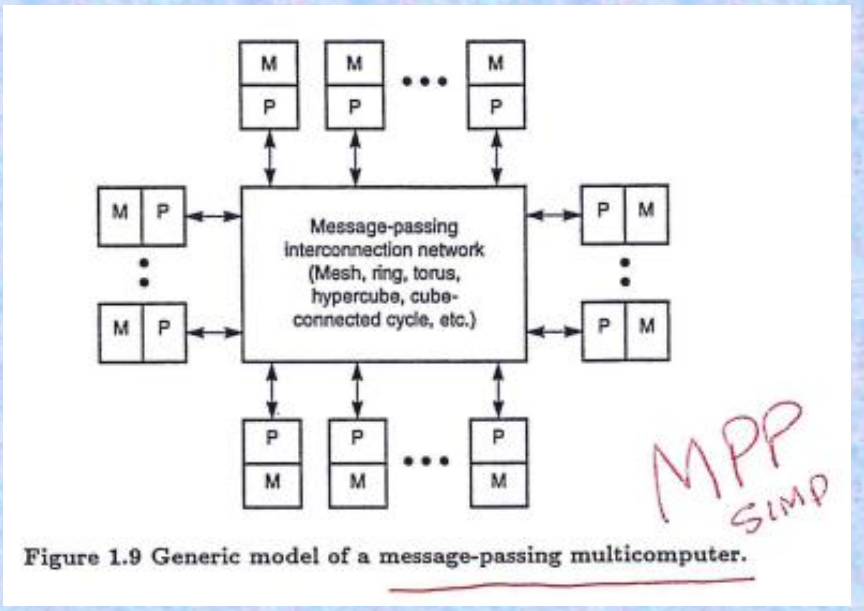


Figure 1.9 Generic model of a message-passing multicomputer.

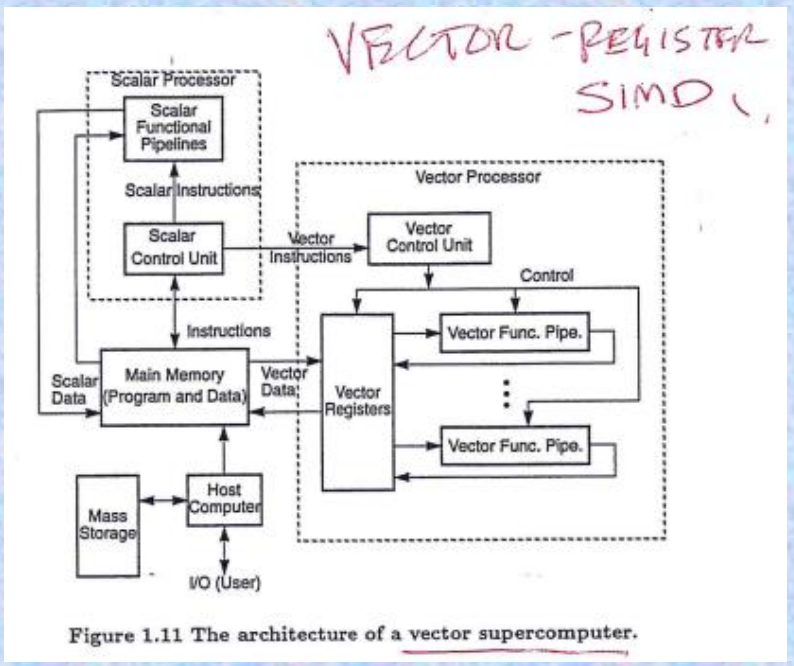


Figure 1.11 The architecture of a vector supercomputer.

K. Hwang, ***" Advanced Computer Architecture: Parallelism, Scalability, Programmability,"*** Boston, MA: McGraw-Hill, 1992.

INTERCONNECT ARCHITECTURES of Supercomputers & Parallel Systems

Learn more in my Lecture on

"INTERCONNECT ARCHITECTURES OF CORES, PROCESSORS, AND NETWORKS"

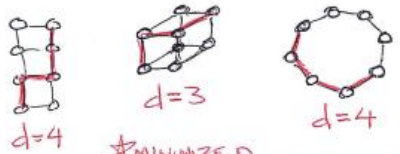
(PDF , MP4 , **YouTube**)

STATIC NETWORK CHARACTERISTICS

N = # OF NODES
★ OPTIMIZE FOR APPLICATION AND TYPICAL PROBLEM GRAIN SIZE

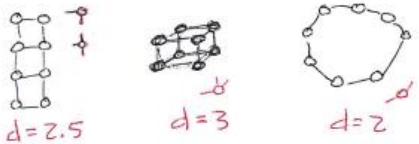
WIRE LENGTH = "CHANNEL" LENGTH = "LINK" LENGTH = "GRAPH EDGE" LENGTH
★ TRY TO MINIMIZE

D = NETWORK DIAMETER
(SHORTEST DISTANCE TO FURTHEST NODE)



★ MINIMIZE D
TO REDUCE COMMUNICATION DELAYS

d = DEGREE
(AVE # OF CONNECTION-CHANNELS INTO NODES)

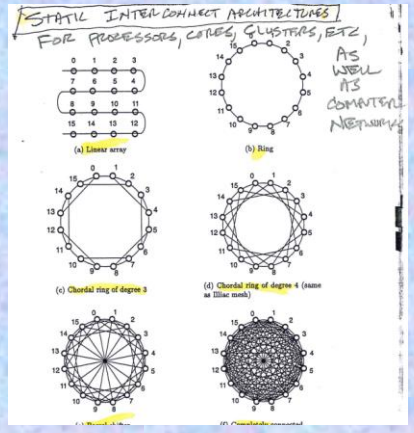


★ A CONSTANT d IS GOOD FOR MODULARITY AND SCALABILITY

B = BISECTION WIDTH
(NUMBER OF LINKS THROUGH SMALLEST CROSS-SECTION)

★ MINIMIZE B TO MINIMIZE REQUIRED WIRING
★ MAXIMIZE B TO MAX NETWORK COMM BANDWIDTH
★ AS B↑, D↓ BUT d↑

W = # OF WIRES IN ONE "CHANNEL", "LINK", "EDGE"



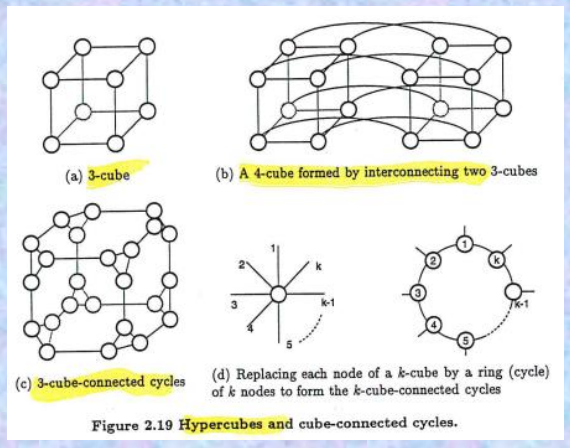


Figure 2.19 Hypercubes and cube-connected cycles.

Table 2.2 Summary of Static Network Characteristics

Network type	Node degree, d	Network diameter, D	No. of links, l	Bisection width, B	Symmetry	Remarks on network size
Linear Array	2	$N-1$	$N-1$	1	No	N nodes
Ring	2	$\lfloor N/2 \rfloor$	N	2	Yes	N nodes
Completely Connected	$N-1$	1	$N(N-1)/2$	$(N/2)^2$	Yes	N nodes
Binary Tree	3	$2(h-1)$	$N-1$	1	No	Tree height $h = \lceil \log_2 N \rceil$
Star	$N-1$	2	$N-1$	$\lfloor N/2 \rfloor$	No	N nodes
2D-Mesh	4	$2(r-1)$	$2N-2r$	r	No	$r \times r$ mesh where $r = \sqrt{N}$
Illiac Mesh	4	$r-1$	$2N$	$2r$	No	Equivalent to a chordal ring of $r = \sqrt{N}$
2D-Torus	4	$2\lfloor r/2 \rfloor$	$2N$	$2r$	Yes	$r \times r$ torus where $r = \sqrt{N}$
Hypercube	n	n	$nN/2$	$N/2$	Yes	N nodes, $n = \log_2 N$ (dimension)
CCC	3	$2k-1+\lfloor k/2 \rfloor$	$3N/2$	$N/(2k)$	Yes	$N = k \times 2^k$ nodes with a cycle length $k \geq$
k-ary n-cube	$2n$	$n\lfloor k/2 \rfloor$	nN	$2k^{n-1}$	Yes	$N = k^n$ nodes

INTERCONNECT ARCHITECTURES of Supercomputers & Parallel Systems

Learn more in my Lecture on

"INTERCONNECT ARCHITECTURES OF CORES, PROCESSORS, AND NETWORKS"

PDF PPTX MP4 **YouTube**

DYNAMIC INTERCONNECT ARCHITECTURES

memory modules or secondary storage devices (disks, tape units, etc.). The system bus is often implemented on a backplane of a printed circuit board. Other boards for processors, memories, or device interfaces are plugged into the backplane board via connectors or cables.

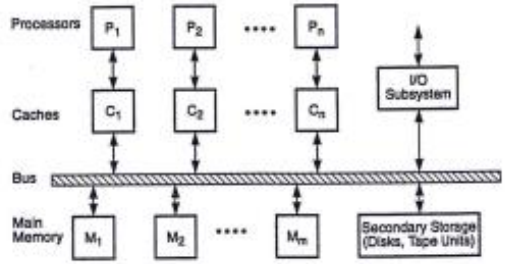
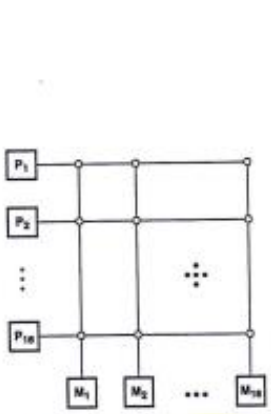
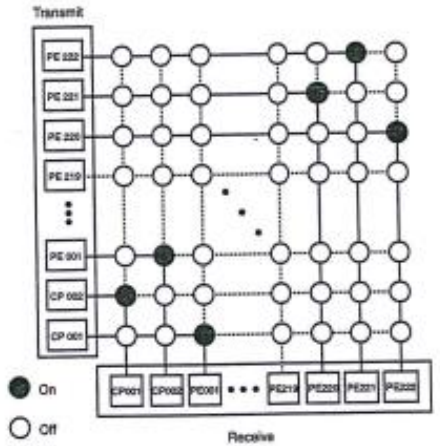


Figure 2.22 A bus-connected multiprocessor system, such as the Sequent Symmetry S1.



(a) Interprocessor-memory crossbar network built in the C.mmp multiprocessor at Carnegie-Mellon University (1972)



(b) The interprocessor crossbar network built in the Fujitsu VPP500 vector parallel processor (1992)

Figure 2.26 Two crossbar switch network configurations.

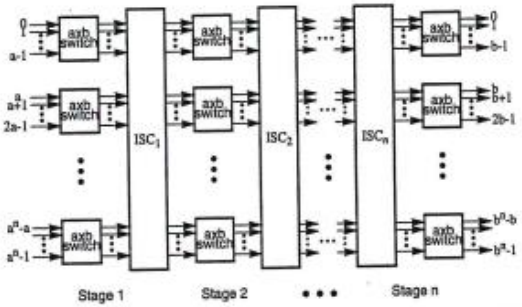


Figure 2.23 A generalized structure of a multistage interconnection network (MIN) built with a × b switch modules and interstage connection patterns $ISC_1, ISC_2, ..., ISC_n$.

Table 2.4 Summary of Dynamic Network Characteristics

Network Characteristics	Bus System	Multistage Network	Crossbar Switch
Minimum latency for unit data transfer	Constant	$O(\log_a n)$	Constant
Bandwidth per processor	$O(w/n)$ to $O(w)$	$O(w)$ to $O(nw)$	$O(w)$ to $O(nw)$
Wiring Complexity	$O(w)$	$O(nw \log_a n)$	$O(n^2 w)$
Switching Complexity	$O(n)$	$O(n \log_a n)$	$O(n^2)$
Connectivity and routing capability	Only one to one at a time.	Some permutations and broadcast, if network unblocked	All permutations, one at a time.
Representative computers	Symmetry S-1, Encore Multimax	BBN TC-2000, IBM RP3	Cray Y-MP/816, Fujitsu VPP500
Remarks	Assume n processors on the bus; bus width is w bits.	$n \times n$ MIN using $k \times k$ switches with line width of w bits.	Assume $n \times n$ crossbar with line width of w bits.

Levels of Computing

1	Embedded
2	PC
3	PC Server or Workstation
4	Mini computer
5	SUPERCOMPUTER

LEVEL	ARCHITECTURE
Embedded	SMP, MPP, or Vector-Register
PC	SMP
PC Server or Workstation	SMP
Mini computer	SMP
SUPERCOMPUTER	SMP, MPP, Vector-Register, *or* Large network dedicated to a single task

LEVEL	APPLICATIONS
Embedded	Real-Time control: Automobiles, Appliances, factory automation, Aerospace
PC	General-purpose “low-end” computing
PC Server or Workstation	LAN server for ~100 people, 3-D simulations, VLSI circuit design
Mini computer	LAN server for ~500 people
SUPERCOMPUTER	SMP: LAN, WAN, or Internet server for 1000’s of people, Air traffic control, NYSE Vector-Register: Matrix-intensive Grand Challenge App’s MPP: Grand Challenge App’s, Chess Large network: Human Genome

LEVEL	CHARACTERISTICS
Embedded	Usually cheap (but not always, like in some defense applications), small, and can be extremely fast – but typically not. May be hardened for industry, space, or defense
PC	Faster than typical embedded, but otherwise relatively slow.
PC Server or Workstation	Faster
Mini computer	Very fast
SUPERCOMPUTER	Extremely fast

LEVEL	EXAMPLE DEVICES
Embedded	• Microcontroller (Intel, Motorola, PIC's) • Microprocessor (Intel, Motorola, PowerPC) • Application Specific IC's (ASIC's) • Programmable Logic Controllers (PLC's)
PC	Microprocessor (Intel, Motorola, PowerPC)
PC Server or Workstation	Multiple microprocessors (Intel, Motorola, PowerPC, Sparc) … Silicon Graphics Terminals, SUN or IBM RS6000 workstations
Mini computer	IBM AS400, Amdahl, HP, Hitachi
SUPERCOMPUTER	SMP: IBM S/390 Vector-Register: CRAY MPP: IBM SP2 Large network: PC's everywhere

LEVEL	OPERATING SYSTEMS
Embedded	None or custom **Possibly a real-time OS**
PC	Windows, DOS, CP/M, OS2, MAC OS, B, Linux, etc.
PC Server or Workstation	Windows NT, UNIX, AIX
Mini computer	UNIX, MVS, VMS, OS 390
Super Computer	SMP: UNIX, MVS, VMS, OS 390 Vector-Register: custom vector OS MPP: custom distributed OS Large network: PC OS's

MICROPROCESSOR	MICROCONTROLLER
For general-purpose computing	Intentionally simple for single-chip embedded applications
Can be **C**omplex **I**nstruction **S**et **C**omputing (CISC)	Intentionally Reduced Instruction Set Computing (RISC)
Both integer and Floating-Point calculations	Intentionally simple integer-only calculations
Large Address Spaces (Plus virtual Addressing)	Can put all data and instructions in on-chip RAM
Versatility	On-chip device control capabilities: DAC, ADC, PWM

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

1996 COMPARISON OF "MINI-COMPUTER" (DR. W. "LEVEL 4") COMPETITORS FOR IBM AS-400 (A SIMPLER CHEAPER VERSION OF IBM S/390

IBM vs. AMDAHL vs. HITACHI

(HDS=Hitachi Data Systems)

A few additions (Y2000 on Pilot, Acquisition Flexibility, etc.). Unlike IBM and Amdahl, HDS makes no statement about Y2K support on Pilot...

11/8/96 G3 vs. Millennium vs. Pilot Feature/Function Comparison

	IBM	Amdahl	HDS
Shipped/References	Yes	Limited	1-3w limited
Performance/LSPR	3.6 - 360	41 - 271	32 - 331
Engines	sub - 10	1 - 8	sub - 10
All CPs "Full Speed"?	Yes	Yes	No (1-3w)
"Quick CP Upgrades"	Yes, x10w	Yes, x2w, 8w	No
Packaging	4, 8 and 12	2, 8	?
Design	microprocessor	bi-polar	microprocessor
Technology	.25 micron cmos	.35 micron cmos	.25 micron cmos
390 CMOS Generation	3	1	1
Key Components Sourced	self	Futitsu	IBM
Integrated DASD	Yes-MP2K	No	No
1997 Upgrade	Yes	No	SOD
PSLC	Yes	Yes	FCS
PSLC/E	Yes	* 1Q97 (PSMF)	1-3w soon(VCMF)
CFCC "L2" microcode	Now	?	?
ICMF Dispatch Assist	Now	?	?
Crypto	2Q97	Channel Attach	4-10W 3Q97
OSA	Now	4Q97 ENTR, 1Q98 for ATM	SOD
VM	Yes	No	Yes
VSE	Yes	SOD	Yes
Battery Backup	Yes	External/Exide	No
CP Sparing	Yes	Yes/dynamic	1Q97
SAP Reassignment	Yes	No/IOP design	????
TCP/IP Assist Instr	Yes	?	Likely
Concur Chnl Upgr/Repair	Yes	Yes	Yes
Dual Power Feed	Yes	3Q97	No
Concurrent LIC Maint	Yes	Yes	Yes
Partial Memory Restart	Yes	No	No
HMC Support	Yes	Yes	Yes
LPAR			
- > 2GB C-Store	Yes	Likely	Likely
- Max LPARs	10	10	10
- Year 2000 Support	Yes	Yes	?
Interpreted SIE	Yes	No	Yes
Cycle Time	6.5, 5.9ns	7.5ns	6.5ns
Min Config	1 Rack	2 Racks	2 Racks
Dual Board	No	Yes	No
List Maintenance	$35/mip/month	$58/mip/month	?
- addt'l for features?	No	?	?
8-way Environmentals	No	Yes	No
- kVA	1.9	5	2.5
- BTUs	6,368	11,700	8,200
- Sq Feet	10.4	16.1	12.9
Channels (High End)			
- Max Total	256	256	256
- Max ESCON	256	256	256
- Max Parallel	96	128	96
- Max OSA	12	16	SOI
- Max Links	16	32	16
Memory (High End)			
- Max System Memory	8G	8G	8G (4-10w)
Acquisition Flexibility			
- lease, purchase	Yes	Yes	Yes
- OSO, EPSO	Yes	No	No

i.e., $100,000 vs. $1,000,000 IN 1996 ☆ DR. W's WORK AT IBM

NOTE: Hitachi Dependent on IBM !!

Notes/Claimed Exclusives

1) The Millennium Dual Board server involves the addition of a second, independent CPU board in the 1st rack. Both CPU boards must share one channel "pool". Amdahl has claimed a "HCD-like" function to move channel groups between boards. These are 2 independent CPU's - not a single "600 mip mainframe".

2) Millennium Global Servers can be converted to Coupling Servers and vice-versa.

3) PSMF on Millennium servers required external coupling links. There is no EMIF capability.

4) Amdahl is claiming the ability to DYNAMICALLY replace a failing CP (1Q97).

5) Amdahl describes a feature called QuickMemory - the ability to upgrade Millennium memory "with minimal downtime". You make the call as to what "minimal downtime" means.

6) Millennium has the following memory increments:
500 Series - 512M, 1G, 1.5G, 2G, 2.5G, 3G, 4G, (6G and 8G/2Q97)
400 Series - 256M, 512M, 768M, 1G, 1.5G, 2G, 2.5G, 3G and 4G

7) Amdahl has announced a new "Max Capping" feature on MDF - "an absolute limit on the amount of CPU time that a domain receives". PR/SM gives IBM customers the ability to weight and cap LPARs.

9) Amdahl has announced that they have "extended the industry compatibility of MDF, making it even easier to manage and operate in a parallel sysplex environment of different processors" This may well mean that the traditional MDF "time slice" mechanisms are being enhanced to be more event driven, or allow independent CP scheduling. Details are non-existent with exception of a caution to users to expect higher overheads.

10) The Battery Backup Feature on Millennium is nothing more than the ability to attach an (external) UPS provided by an alliance with Exide Electronics.

11) The Millennium "High Speed Host Security Module (HSM)" continues a feature Amdahl has had on their 5995M Family to channel attach an encryption engine provided via an alliance with Racal Data Group.

12) HDS re-stated Pilot performance to be "model for model equivalent to IBM's G3". However, Pilot 1-3 way models come equipped with half the cache and buses... HDS has been quietly backing off equivalence for these models.

13) Pilot 4-10way models require from one to 3 I/O expansion cages. One comes standard, a second is required for > 96 channels, and a third for configurations with more than 176 channels. 1-3ways require an expansion cage with more than 48 channels.

Microprocessors (with Floating-Point) vs. Microcontrollers (only Integers)

INTEGER NUMBER RANGES

```
8-bit unsigned:      0    to  (2^8)-1     =       0 to 255
8-bit signed:  -(2^8)/2 to ((2^8)/2)-1 =    -128 to 127

16-bit unsigned:     0    to (2^16)-1     =       0 to 65,535
16-bit signed: -(2^16)/2to((2^16)/2)-1 = -32,768to 32,767

32-bit unsigned:       0 to (2^32)-1     =       0 to 4,294,967,295
32-bit signed: -(2^32)/2to((2^32)/2)-1 = -2,147,483,648
                                           to  2,147,483,647
n-bit unsigned:        0 to  (2^n)-1
n-bit signed:  -(2^n)/2 to  ((2^n)/2)-1
```

FLOATING POINT NUMBER RANGES

```
IEEE single precision (32-bit) BFP -1*10^38 to 1*10^38
IEEE double precision (64-bit) BFP -1*10^308 to 1*10^308
```

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

Microprocessors (with Floating-Point) vs. Microcontrollers (only Integers)

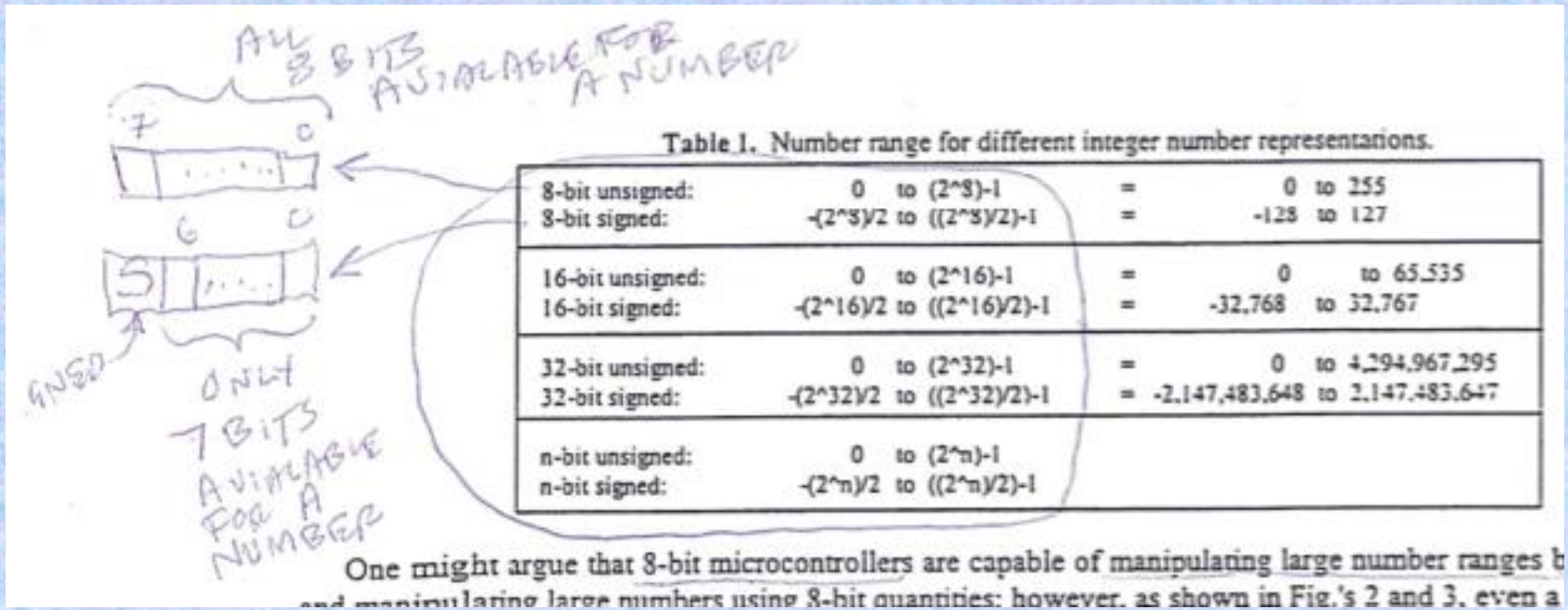


Table 1. Number range for different integer number representations.

8-bit unsigned:	0 to (2^8)-1	=	0 to 255
8-bit signed:	-(2^8)/2 to ((2^8)/2)-1	=	-128 to 127
16-bit unsigned:	0 to (2^16)-1	=	0 to 65,535
16-bit signed:	-(2^16)/2 to ((2^16)/2)-1	=	-32,768 to 32,767
32-bit unsigned:	0 to (2^32)-1	=	0 to 4,294,967,295
32-bit signed:	-(2^32)/2 to ((2^32)/2)-1	=	-2,147,483,648 to 2,147,483,647
n-bit unsigned:	0 to (2^n)-1		
n-bit signed:	-(2^n)/2 to ((2^n)/2)-1		

One might argue that 8-bit microcontrollers are capable of manipulating large number ranges b
and manipulating large numbers using 8-bit quantities; however, as shown in Fig.'s 2 and 3, even a

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

Microprocessors (with Floating-Point) vs. Microcontrollers (only Integers)

INTEGER NUMBER PRECISION (i.e., smallest number)

8-bit unsigned:	**1**
8-bit signed:	**1**
16-bit unsigned:	**1**
16-bit signed:	**1**
32-bit unsigned:	**1**
32-bit signed:	**1**
n-bit unsigned:	**1**
n-bit signed:	**1**

FLOATING POINT NUMBER PRECISION

IEEE single precision (32-bit) BFP: **23-bit MANTISSA (right of decimal point)**
IEEE double precision (64-bit) BFP: **52-bit MANTISSA (right of decimal point)**

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

Review of how Floating-Point Works

Taught in Introductory Computer Science Courses (e.g. CS/EGR 230)

"Number and Character Representations in Computing"
PDF PPTX-w/audio MP4 **YouTube**

"An example of IEEE Binary Floating-Point (BFP)"
PDF PPTX-w/audio MP4 **YouTube**

Recall J. Wunderlich “Minimal Computer Architecture”

Taught in prerequisite Course *EGR/CS332* or *EGR330*

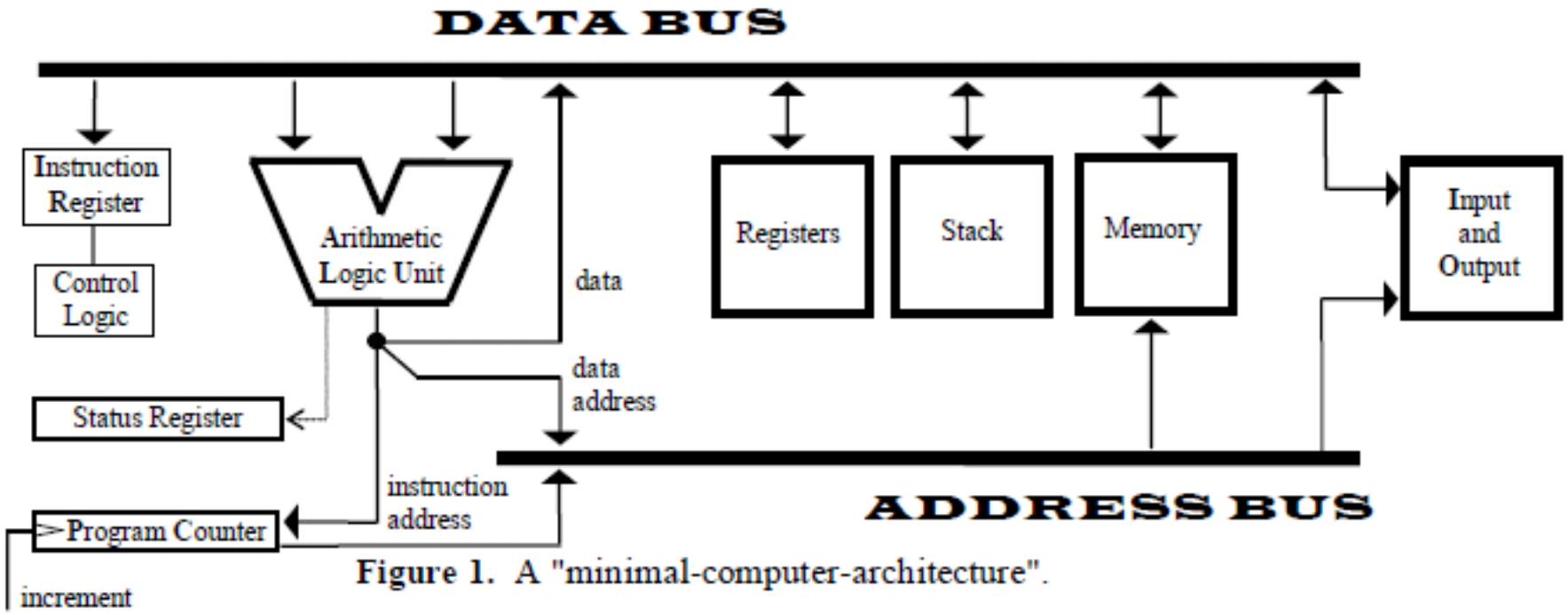


Figure 1. A "minimal-computer-architecture".

- A program counter to address instructions to be fetched from memory.
- An instruction register to put the fetched instruction in.
- Control logic to create all routing signals after decoding the fetched instruction.
- An ALU for arithmetic and logical manipulation of data and addresses.
- Registers for storing intermediate results of calculations.
- A status register for status flags and condition codes.
- Memory for storing data and instructions.
- A stack for storing addresses (or processor status) for returning from program-calls (or interrupts).
- I/O which is addressed as memory (i.e., memory-mapped I/O).

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

Microprocessors vs. Microcontrollers

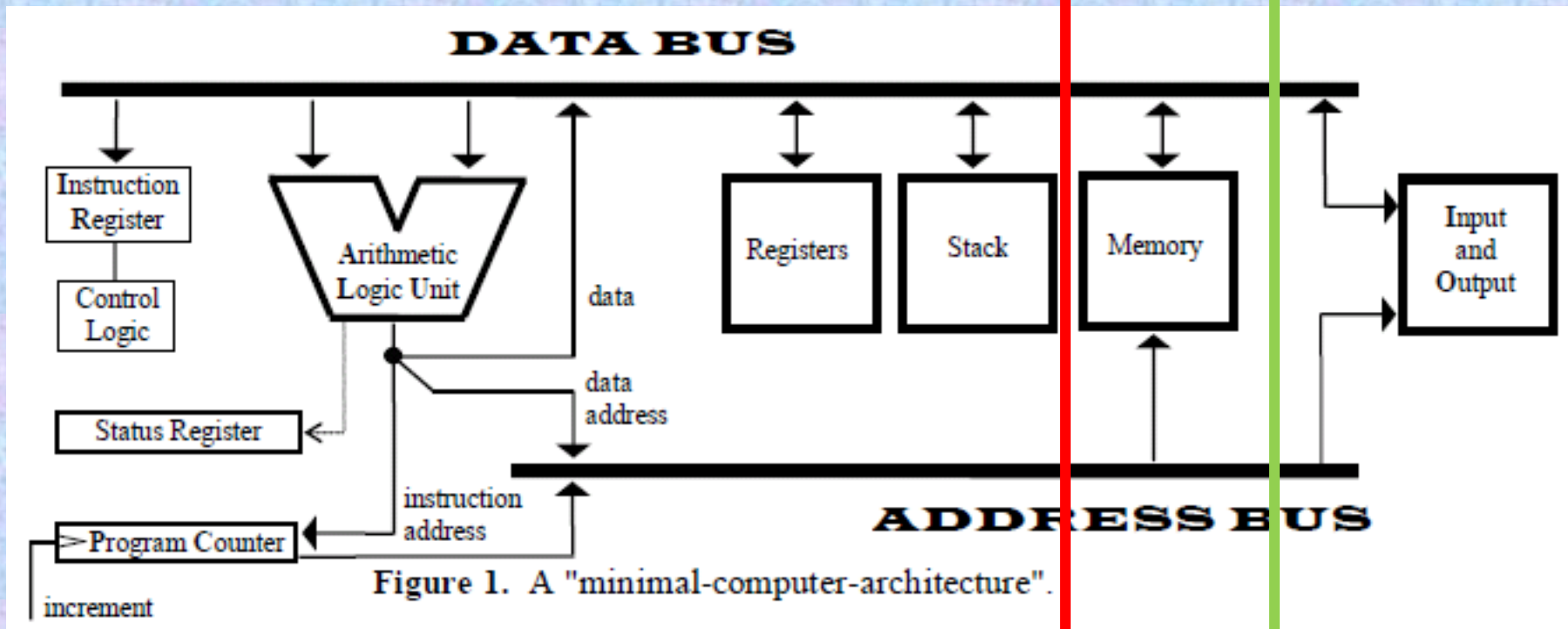


Figure 1. A "minimal-computer-architecture".

Microprocessors have several Cache's between CPU and "Memory"

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

Microprocessors

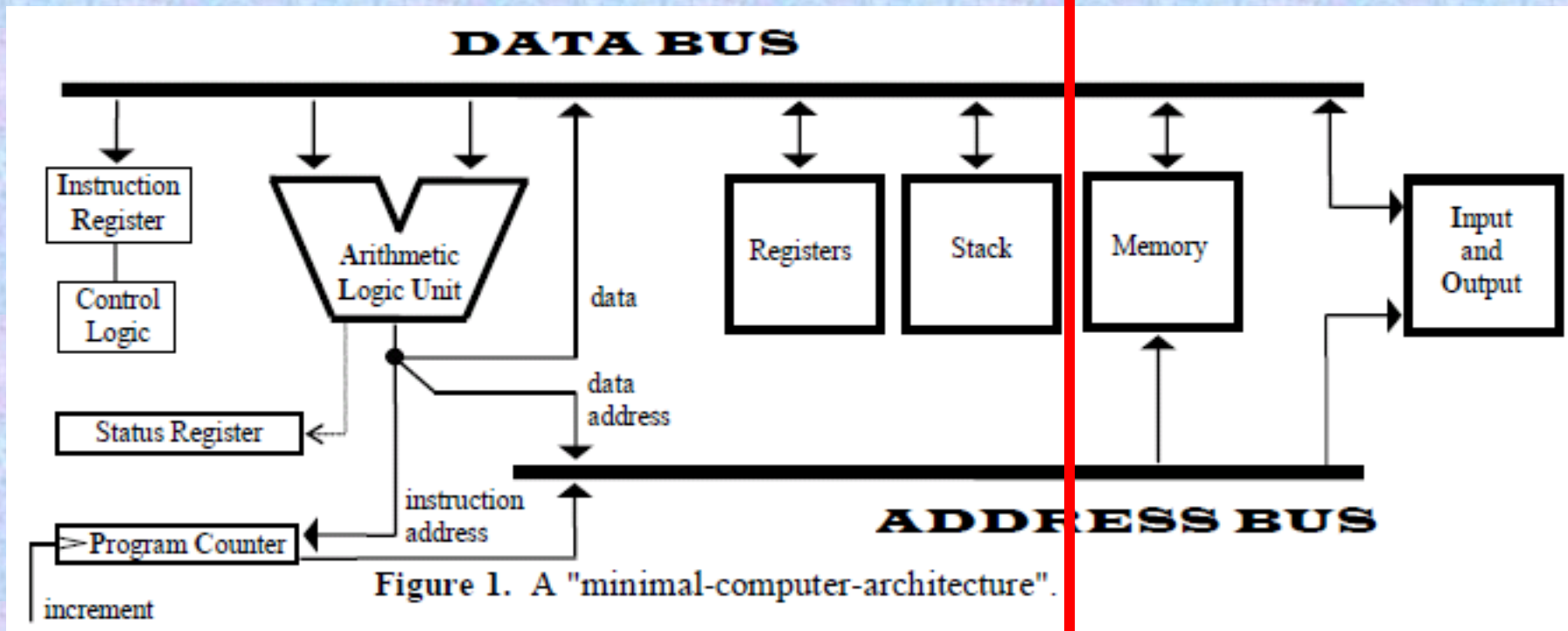


Figure 1. A "minimal-computer-architecture".

Microprocessors have several Cache's between CPU and "Memory"

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

Microprocessors and SUPERCOMPUTERS have several Cache's between CPU's/Cores and "Memories"

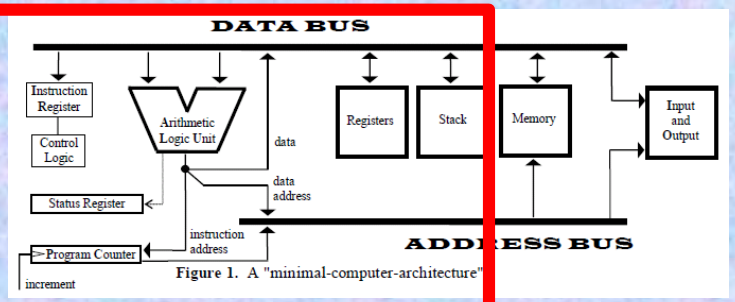
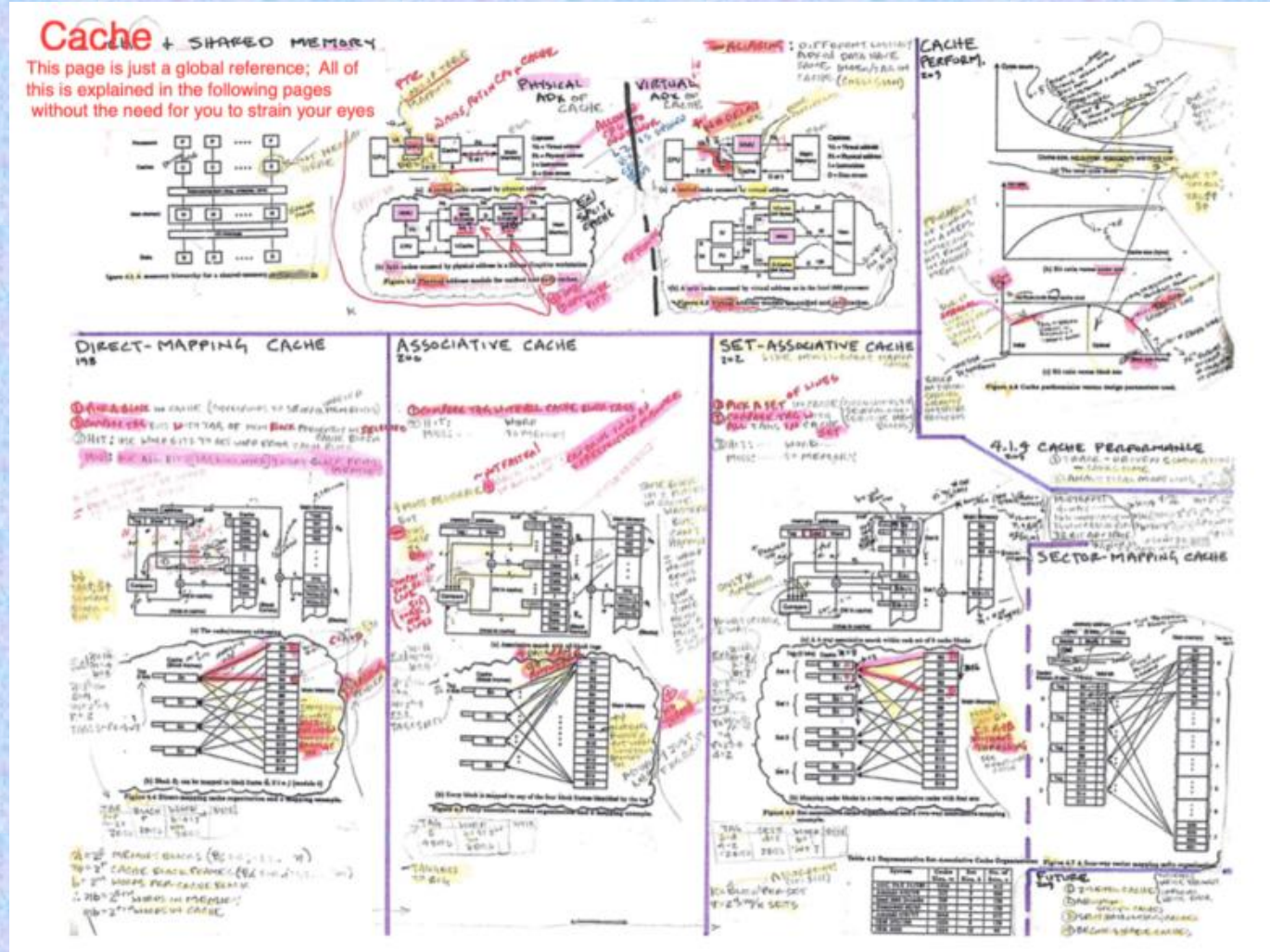


Figure 1. A "minimal-computer-architecture"

Learn more in my lecture(s) on CACHE, later in EGR/CS433 / EGR430):
"CACHE DESIGN" (PPTX-w/audio,PDF,**MP4**,**YouTube**)



Consider the time it takes to execute a segment of code

$$T = \overline{CPI} * (I_c) * \tau$$

K. Hwang, *"Advanced Computer Architecture: Parallelism, Scalability, Programmability"*, McGraw-Hill, 1993.

where τ is the clock period in seconds per cycle (i.e., 1/*frequency*), I_c is the number of machine instructions in a given code segment, and $\overline{CPI}$ (cycles per instruction) is the average time to fetch, decode, execute, and store results for each instruction [5]. There are many strategies to decrease $\overline{CPI}$; for example, processing several instructions simultaneously (i.e., superscalar), or moving data directly between I/O and memory (i.e., Direct Memory Access). Hardware to anticipate and take "*pre*-actions" has been a design concept for many years. This not only includes prefetching data and instructions in caches, but also prefetching branch-target addresses using *Branch History Tables*, or *caching* virtual address translations using *Translation Lookaside Buffers*. Other speed-up techniques include re-ordering and optimizing instruction streams as they come into the CPU (i.e., out-of-order execution), or overlapping the individual instruction-cycle phases of many instructions (i.e., super-pipelined).

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

Considering the time it takes to execute a segment of Machine code

$$T = \overline{CPI} * (I_c) * \tau$$

Dictated by computer architects developing new microprocessors microcontrollers, supercomputer, etc.

Dictated by engineers and programmers in how they code (e.g., High-level vs. Assembly) or in what software they choose

Dictated by device physicists and material scientists developing faster-switching transistors

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

EXAMPLE for ROBOTICS and AEROSPACE/DEFENSE

Simulation	Real-Time control
Using good engineering and physics, create a model of a physical system (i.e., not just a cartoon)	Establish stable closed loop control with a good model (“Plant”) that represents physical system being controlled
Vary inputs to simulation to better understand model	Fine tune PID control to better manipulation of physical system
Use more complex computer hardware to enhance graphics and model complexity	**Intentionally simplify** all hardware to yield fast, compact, fault-tolerant, real-time responses
Use more complex computer software to enhance graphics and minimize programming effort	**Intentionally simplify** code to yield fast, compact, fault-tolerant, real-time responses. No operating system or a real-time OS may be best
Interact with real-time code to improve physical model and build ENVIRONMENTAL MAPS	Interact with simulation to obtain GLOBAL PATH-PLANNING rather than Local

The evolution of microprocessors (as well as more complex high-performance machines) has led to many advances in computer architecture to speed-up processing; this has included much more than increasing processor clock speed. The time to execute a program can be represented by:

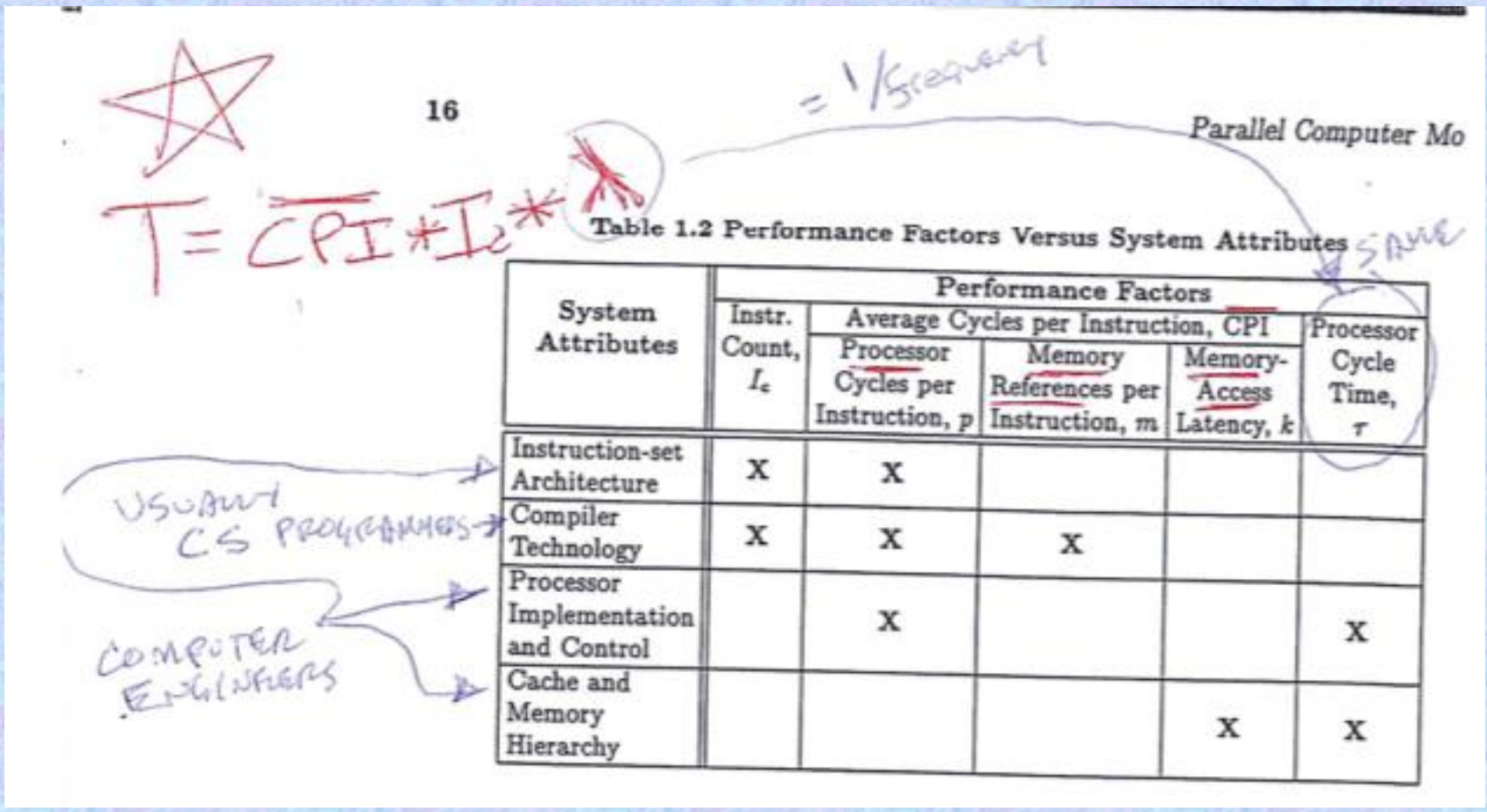
$$T = \overline{CPI} * (I_c) * \lambda$$

where λ is the clock period in seconds per cycle (i.e., 1/*frequency*), I_c is the number of machine instructions in a given code segment, and $\overline{CPI}$ (cycles per instruction) is the average time to fetch, decode, execute, and store results for each instruction [5]. There are many strategies to decrease $\overline{CPI}$; for example, processing several instructions simultaneously (i.e., superscalar), or moving data directly between I/O and memory (i.e., Direct Memory Access). Hardware to anticipate and take "*pre*-actions" has been a design concept for many years. This not only includes prefetching data and instructions in caches, but also prefetching branch-target addresses using *Branch History Tables*, or *caching* virtual address translations using *Translation Lookaside Buffers*. Other speed-up techniques include re-ordering and optimizing instruction streams as they come into the CPU (i.e., out-of-order execution), or overlapping the individual instruction-cycle phases of many instructions (i.e., super-pipelined). Many of these advances will eventually work their way into microcontroller architectures, however features designed to handle large address spaces are less likely to be needed for microcontroller applications.

Although both devices have an ALU for integer arithmetic and logical manipulation of data, microprocessors are much better suited for "number-crunching", and usually have a *wider* data bus and *larger* general-purpose registers to accommodate this. Microcontrollers are typically limited to 8-bit or 16-bit number representations (even though 32-bit microcontrollers are available); whereas microprocessors usually allow 32-bit representations, and contain additional floating-point hardware to allow arithmetic using much larger number ranges (and therefore much greater precision). Table 1 shows the available number range for different integer number representations.

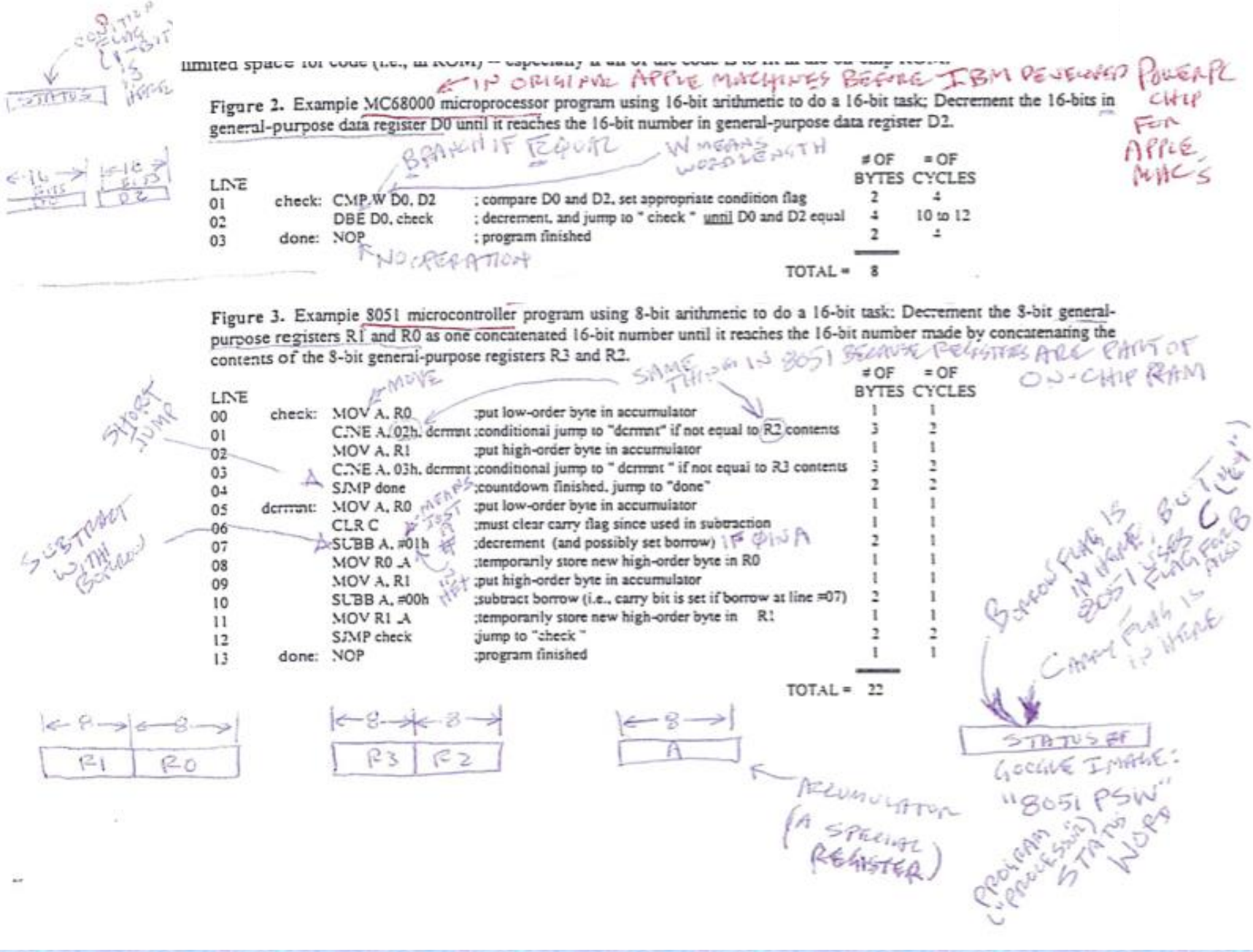
CPI = P + (m K)
WHERE P = # OF CYCLES FOR INSTRUCT DECODE + EXEC
m = # OF MEM ACCESSES PER INSTR
K = RATIO OF MEM ACCESS CYCLE TIME TO PROC DECODE AND EXECUTE TIME
(IE: HOW MUCH SLOWER TO GET THINGS FROM MEMORY THAN TO DO THINGS IN PROC

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.



System Attributes	Performance Factors				
	Instr. Count, I_c	Average Cycles per Instruction, CPI			Processor Cycle Time, τ
		Processor Cycles per Instruction, p	Memory References per Instruction, m	Memory-Access Latency, k	
Instruction-set Architecture	X	X			
Compiler Technology	X	X	X		
Processor Implementation and Control		X			X
Cache and Memory Hierarchy				X	X

K. Hwang, ***" Advanced Computer Architecture: Parallelism, Scalability, Programmability,"*** Boston, MA: McGraw-Hill, 1992.



limited space for code (i.e., in ROM) -- especially if all of the code is to fit in the on-chip ROM

Figure 2. Example MC68000 microprocessor program using 16-bit arithmetic to do a 16-bit task; Decrement the 16-bits in general-purpose data register D0 until it reaches the 16-bit number in general-purpose data register D2.

LINE				# OF BYTES	= OF CYCLES
01	check:	CMP.W D0, D2	; compare D0 and D2, set appropriate condition flag	2	4
02		DBE D0, check	; decrement, and jump to " check " until D0 and D2 equal	4	10 to 12
03	done:	NOP	; program finished	2	4
			TOTAL =	8	

Figure 3. Example 8051 microcontroller program using 8-bit arithmetic to do a 16-bit task: Decrement the 8-bit general-purpose registers R1 and R0 as one concatenated 16-bit number until it reaches the 16-bit number made by concatenating the contents of the 8-bit general-purpose registers R3 and R2.

LINE				# OF BYTES	= OF CYCLES
00	check:	MOV A, R0	;put low-order byte in accumulator	1	1
01		CJNE A, 02h, dcrmnt	;conditional jump to "dcrmnt" if not equal to R2 contents	3	2
02		MOV A, R1	;put high-order byte in accumulator	1	1
03		CJNE A, 03h, dcrmnt	;conditional jump to " dcrmnt " if not equal to R3 contents	3	2
04		SJMP done	;countdown finished, jump to "done"	2	2
05	dcrmnt:	MOV A, R0	;put low-order byte in accumulator	1	1
06		CLR C	;must clear carry flag since used in subtraction	1	1
07		SUBB A, #01h	;decrement (and possibly set borrow)	2	1
08		MOV R0, A	;temporarily store new high-order byte in R0	1	1
09		MOV A, R1	;put high-order byte in accumulator	1	1
10		SUBB A, #00h	;subtract borrow (i.e., carry bit is set if borrow at line #07)	2	1
11		MOV R1, A	;temporarily store new high-order byte in R1	1	1
12		SJMP check	;jump to "check "	2	2
13	done:	NOP	;program finished	1	1
			TOTAL =	22	

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

One strength of microcontrollers is their on-chip RAM which allows faster memory access and therefore fewer cycles per instruction. This is illustrated in Fig.'s 4 and 5. Although the MC68000 microprocessor code of Fig. 4 requires less bytes, it must access the off-chip RAM for reading and writing the initial and final *count*; this is significantly slower than manipulating on-chip RAM. However, the overall speed of the MC68000 microprocessor code in Fig. 4 would be as fast as the 8051 microcontroller code of Fig. 5 if the difference between the initial *count* and the desired *count* was large enough (assuming equal clock speeds). The above discussion can be easily extended to a comparison of 16-bit and 32-bit devices (i.e., when doing 32-bit arithmetic).

Figure 4. Example MC68000 microprocessor program using 16-bit arithmetic to do a 16-bit task; Decrement the 16-bits at RAM location 2000h until it reaches the 16-bit number in general-purpose data register D2; then store count back into memory.

LINE				# OF BYTES	# OF CYCLES
00		MOVE.W $2000, D0	; copy original count into register D0 from RAM (off-chip)	4	12
01	check:	CMP.W D0, D2	; compare D0 and D2, set appropriate condition flag	2	4
02		DBE D0, check	; decrement, and jump to " check " until D0 and D2 equal	4	10 to 12
03		MOVE.W D0, $2000	; write count to RAM (off-chip) from D0	4	12
04	done:	NOP	; program finished	2	4
			TOTAL =	16	42 to 44

Figure 5. Example 8051 microcontroller program using 8-bit arithmetic to do a 16-bit task; Decrement the 8-bit contents of internal RAM addresses 21h and 20h as one concatenated 16-bit number until it reaches the 16-bit number made by concatenaring the contents of the 8-bit general-purpose registers R3 and R2 [2].

LINE				# OF BYTES	# OF CYCLES
00	check:	MOV A, 20h	;get low-order byte from on-chip RAM	2	1
01		CJNE A, 02h, dcrmnt	;conditional jump to "dcrmnt" if not equal to R2 contents	3	2
02		MOV A, 21h	;get high-order byte from on-chip RAM	2	1
03		CJNE A, 03h, dcrmnt	;conditional jump to " dcrmnt " if not equal to R3 contents	3	2
04		SJMP done	;countdown finished, jump to "done"	2	2
05	dcrmnt:	MOV A, 20h	;get low-order byte from on-chip RAM for decrementing	2	1
06		CLR C	;must clear carry flag since it is used as a borrow	1	1
07		SUBB A, #01h	;decrement (and possibly set borrow)	2	1
08		MOV 20h, A	;store new high-order byte in on-chip RAM	2	1
09		MOV A, 21h	;get high-order byte from on-chip RAM for decrementing	2	1
10		SUBB A, #00h	;subtract borrow (i.e., carry bit is set if borrow at line #07)	2	1
11		MOV 21h, A	;store new low-order byte in on-chip RAM	2	1
12		SJMP check	;jump to "check "	2	2
13	done:	NOP	;program finished	1	1
			TOTAL =	28	18

MUCH FASTER THAN

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

The Program Status Word (PSW) is not just a “Status Register”!

8051 FLAG BITS AND THE PSW REGISTER

- **PSW:** It is 8 bits wide but use only 6 bits .There are 4 conditional flags (CY,AC,P & OV) , 2 user-definable flags (PSW.1 & PSW.5) and 2 register bank selector(RS1 & RS0).

CY	AC	F0	RS1	RS0	OV	– –	P

CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary carry flag.
F0	PSW.5	Available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1.
RS0	PSW.3	Register Bank selector bit 0.
OV	PSW.2	Overflow flag.
--	PSW.1	User definable bit.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of 1 bits in the accumulator.

RS1	RS0	Register Bank	Address
0	0	0	00H – 07H
0	1	1	08H – 0FH
1	0	2	10H – 17H
1	1	3	18H – 1FH

PSW is not just a “Status Register”! – it CONFIGURES MACHINE !!

- Especially in Supercomputer's & Large-Scale Servers !!

See IBM S/390 (“Z/Architecture in this Millennium”) PSW:

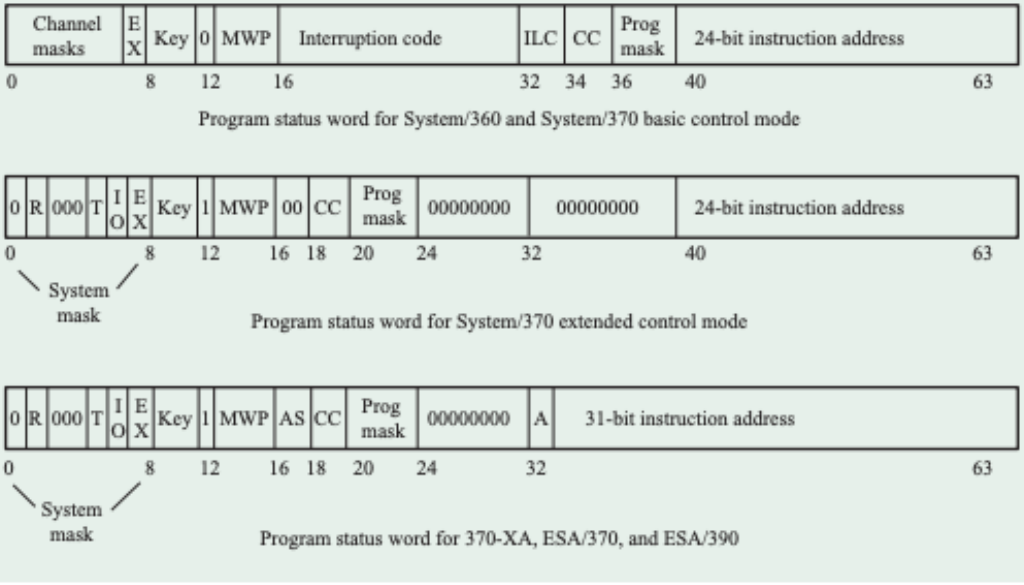


Figure 2

PSW formats for System/360 through ESA/390.

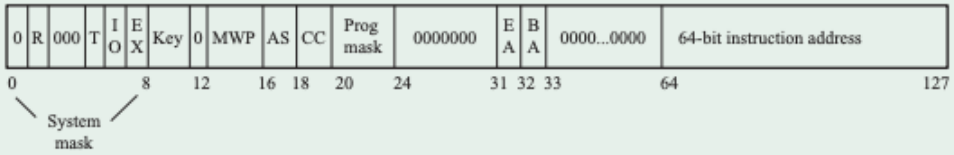
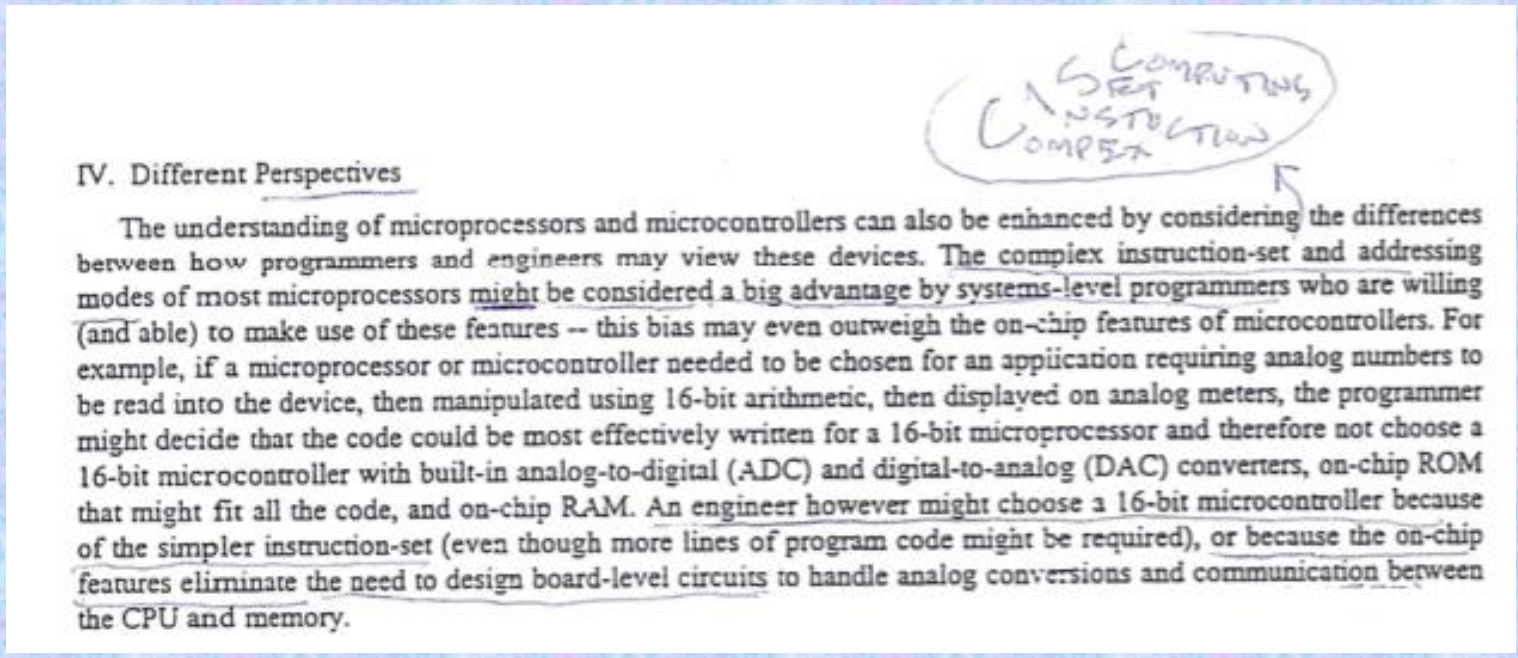


Figure 7

PSW format for z/Architecture.

https://www.researchgate.net/figure/RXE-instruction-format_fig1_224103149

CISC vs. RISC



IV. Different Perspectives

The understanding of microprocessors and microcontrollers can also be enhanced by considering the differences between how programmers and engineers may view these devices. The complex instruction-set and addressing modes of most microprocessors might be considered a big advantage by systems-level programmers who are willing (and able) to make use of these features -- this bias may even outweigh the on-chip features of microcontrollers. For example, if a microprocessor or microcontroller needed to be chosen for an application requiring analog numbers to be read into the device, then manipulated using 16-bit arithmetic, then displayed on analog meters, the programmer might decide that the code could be most effectively written for a 16-bit microprocessor and therefore not choose a 16-bit microcontroller with built-in analog-to-digital (ADC) and digital-to-analog (DAC) converters, on-chip ROM that might fit all the code, and on-chip RAM. An engineer however might choose a 16-bit microcontroller because of the simpler instruction-set (even though more lines of program code might be required), or because the on-chip features eliminate the need to design board-level circuits to handle analog conversions and communication between the CPU and memory.

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

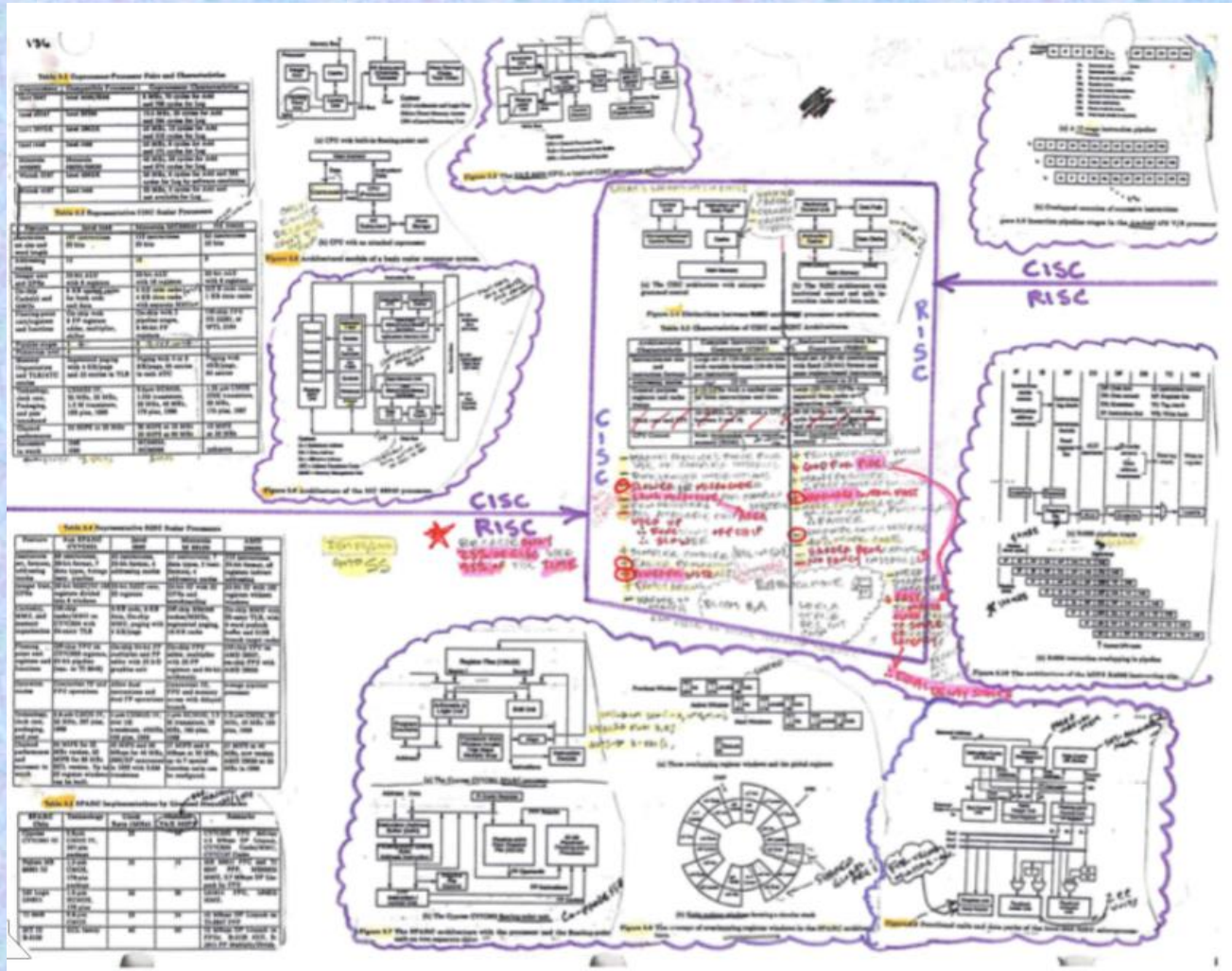
IBM S/390 ("Z/Architecture in this Millennium") have approx. 2000 machine instructions, whereas microprocessors typically have approx. 800 instructions, and microcontrollers between 32 and ~250 instructions

CISC vs. RISC

Learn more in my lecture(s) on CISC vs. RISC, later in EGR/CS433 / EGR430):

"CISC vs. RISC INSTRUCTION SET DESIGN, PERFORMANCE & MULTIPROCESSOR SCALABILITY "

PPTX-w/audio PDF **MP4** **YouTube**



Also, Microprocessors and Supercomputers need VIRTUAL ADDRESSING
i.e., CPU and all Software uses 64-bit Addresses' (2^64 = 16 Billion Billion) even though Physical Memory much Smaller!! (e.g. 2^40cpu_pins_for_RAM = a Terabyte of RAM)

IBM S/390 ("Z/Architecture in this Millennium")

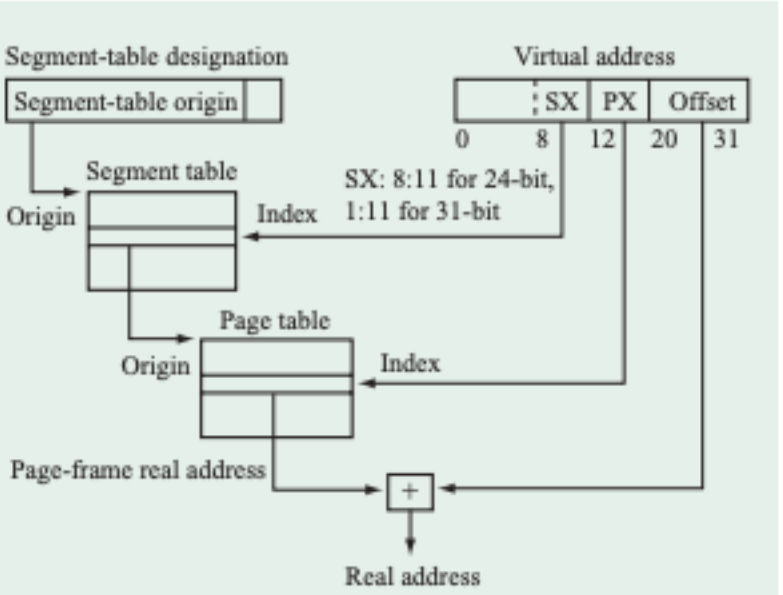


Figure 3

Dynamic address translation for System/370 through ESA/390.

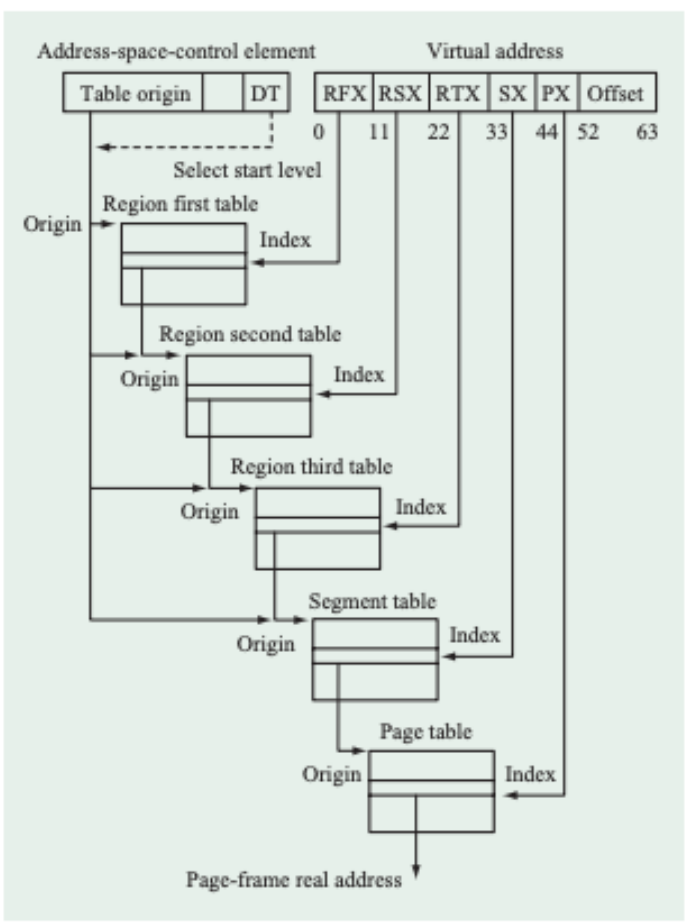


Figure 6

z/Architecture dynamic address translation.

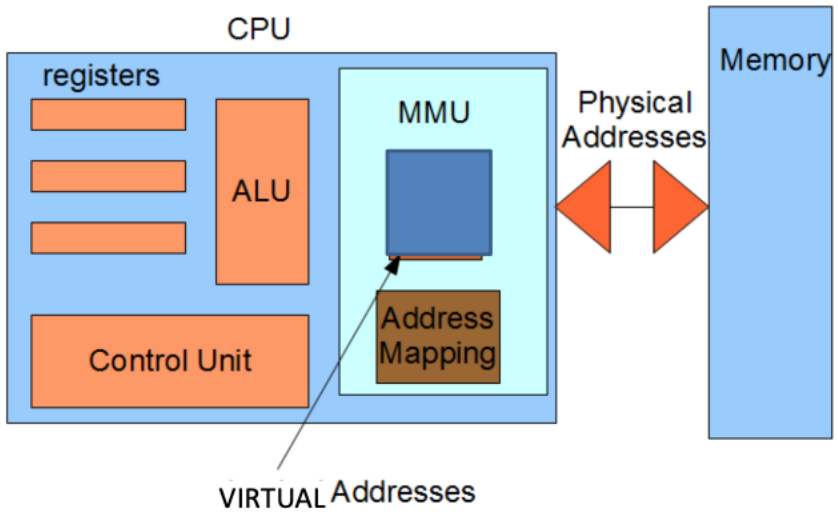
https://www.researchgate.net/figure/RXE-instruction-format_fig1_224103149

Also, Microprocessors and Supercomputers need VIRTUAL ADDRESSING
i.e., CPU and all Software uses 64-bit Addresses'
even though Physical Memory much Smaller!!

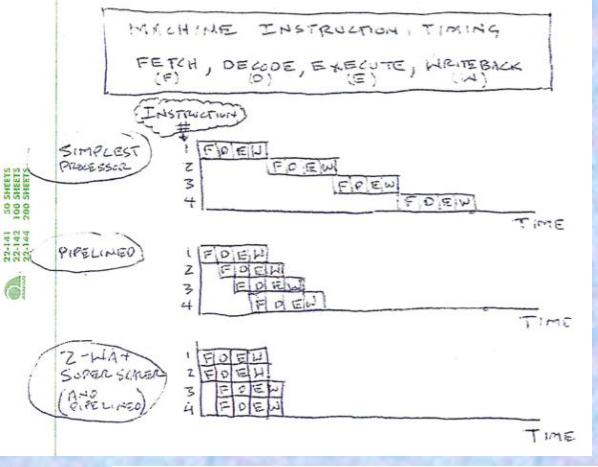
Learn more in my lecture(s) later in EGR/CS433 / EGR430):
"VIRTUAL MEMORY" (PPTX-w/audio,PDF,**MP4**,**YouTube**)

VIRTUAL ADDRESSES --> PHYSICAL ADDRESSES

The CPU, operating system, and all application programs use a **64-bit** address space (VIRTUAL addressing); but 2^64 is ~16,000,000,000 GigaBytes (i.e., 2^4 x 2^30 x 2^30) of memory which is much more than the Physical Memory of most computers. For example the motherboard of most PCs, and the number of address pins coming out of the processor is typically only 40 which corresponds to 2^4**0** = 1 TeraByte (i.e., 2^10 x 2^30 = 1000 GigaBytes)



Before memory addresses are loaded on to the system bus, they are translated to physical addresses by the MMU.



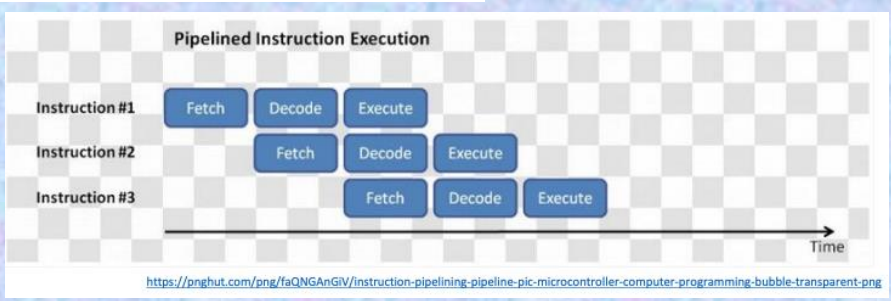
Review

Instruction PIPELINES ("Machine Cycles") and INSTRUCTION FORMATS work

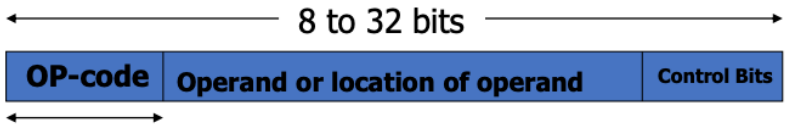
Taught in prerequisite Course(s)

EGR/CS332 or *EGR330*

1. CPU Pipeline
2. CPU Design



Typical Instruction Format:



5 to 11 bits in OP-Code

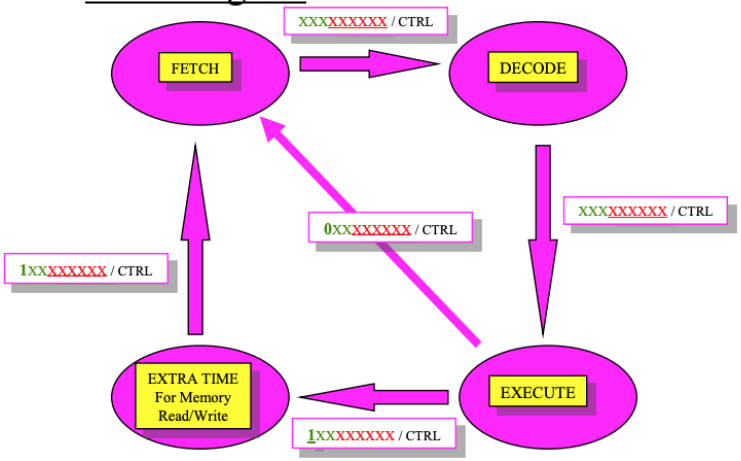
Therefore 2^5=32 to 2^11=2048 different machine instructions in "**Instruction Set**"

- Some simple Microcontrollers (e.g., PIC's) have only 32 instructions
- Some large-scale machines (e.g., IBM S/390) have close to 2000 instructions

Step 2

NOTE:
CNTRL = all control signals to *DATA PATH*

- Create State Diagram

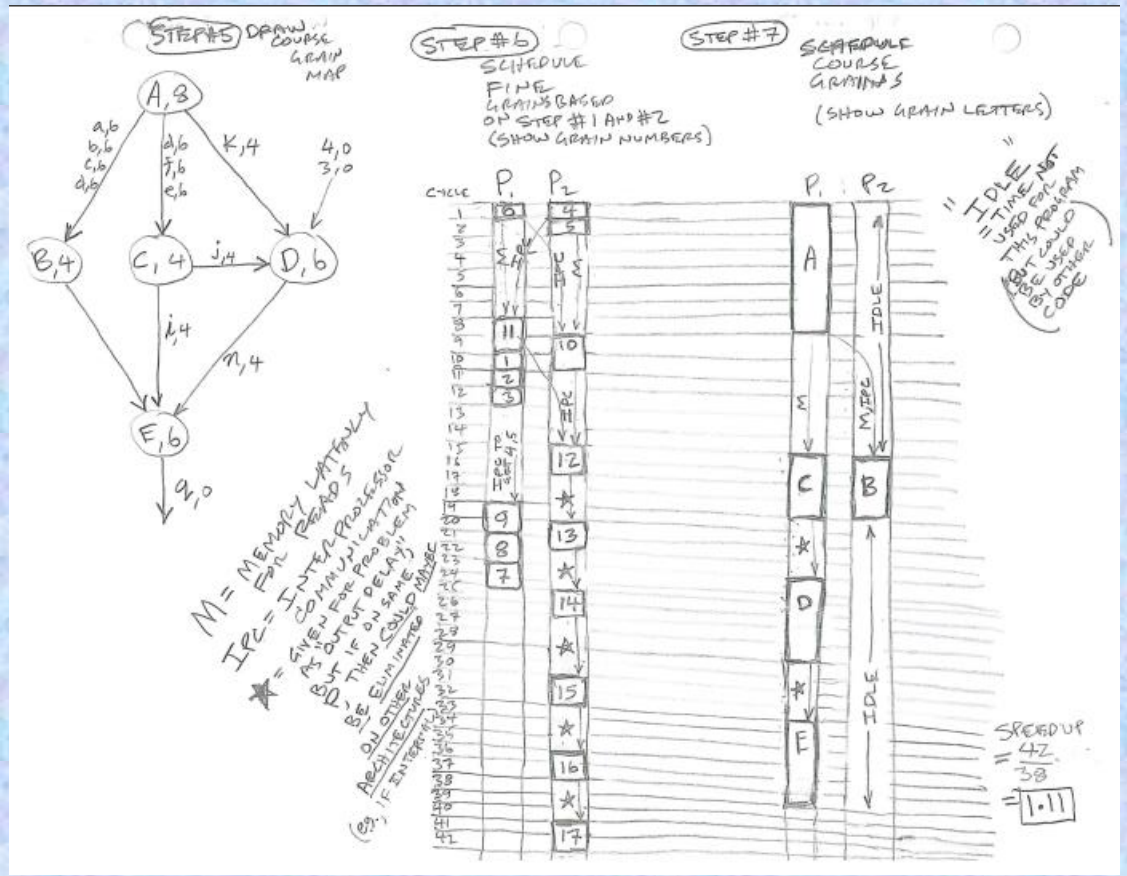


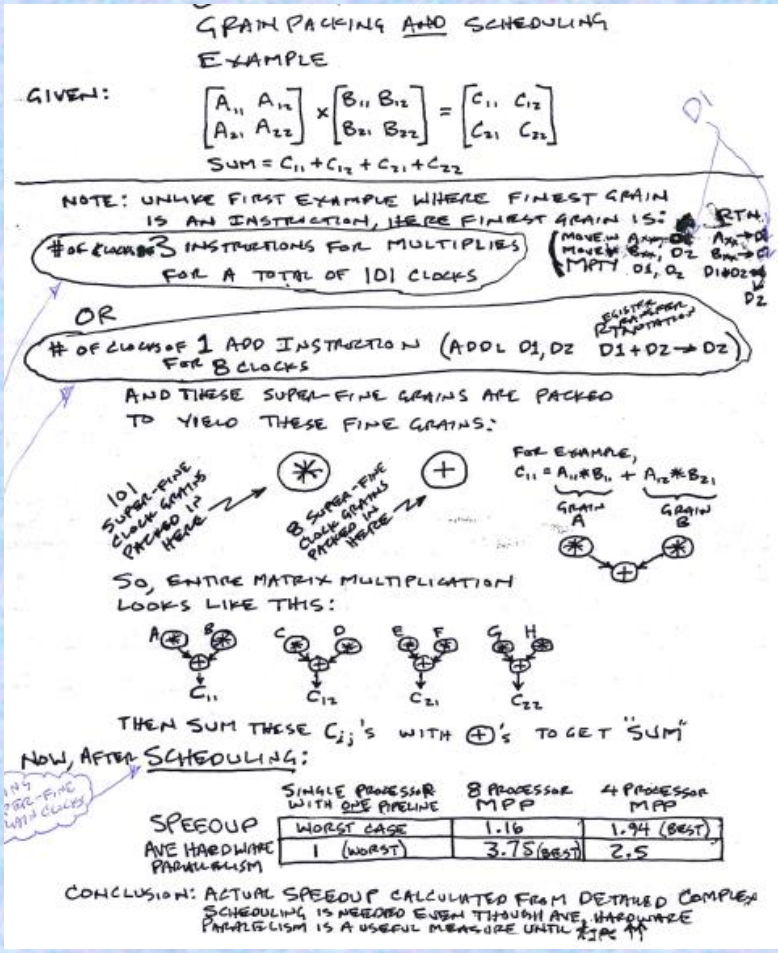
Parallel processing systems now need out of order execution, and optimized scheduling to avoid penalties where dependencies between data, I/O, and Control can cause delays when one part is waiting on another.

Learn more in my later lecture(s) in EGR/CS433 / EGR430:

"PARALLEL PROCESSING FUNDAMENTALS"

PDF PPTX **MP4** **YouTube**





	SINGLE PROCESSOR WITH ONE PIPELINE	8 PROCESSOR MPP	4 PROCESSOR MPP
SPEEDUP	WORST CASE	1.16	1.94 (BEST)
AVE HARDWARE PARALLELISM	1 (WORST)	3.75 (BEST)	2.5

Introductory Lab Project

Figure 6 shows a digital-design course laboratory project to introduce the control and flow of data in a microprocessor or microcontroller. The counters are analogous to timers in a microcontroller or general-purpose registers used as counters in a microprocessor. The select line to the multiplexer can represent a *control-logic* signal generated after decoding the op-code; the comparator can represent a simplified arithmetic logic unit (ALU); and the L.E.D. circuits controlled by the comparator output can represent the contents of a status register. A variation of this lab can be made by replacing the comparator with a 2-bit parallel adder.

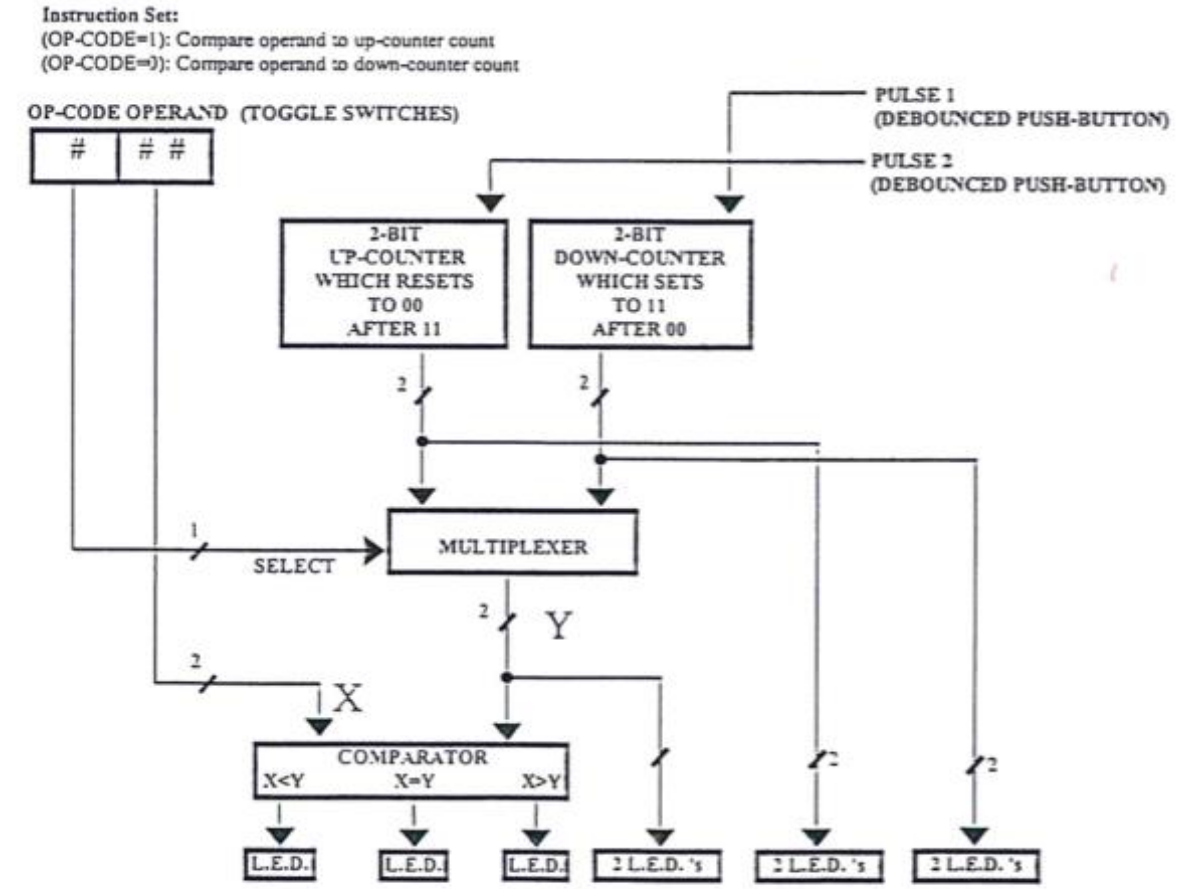
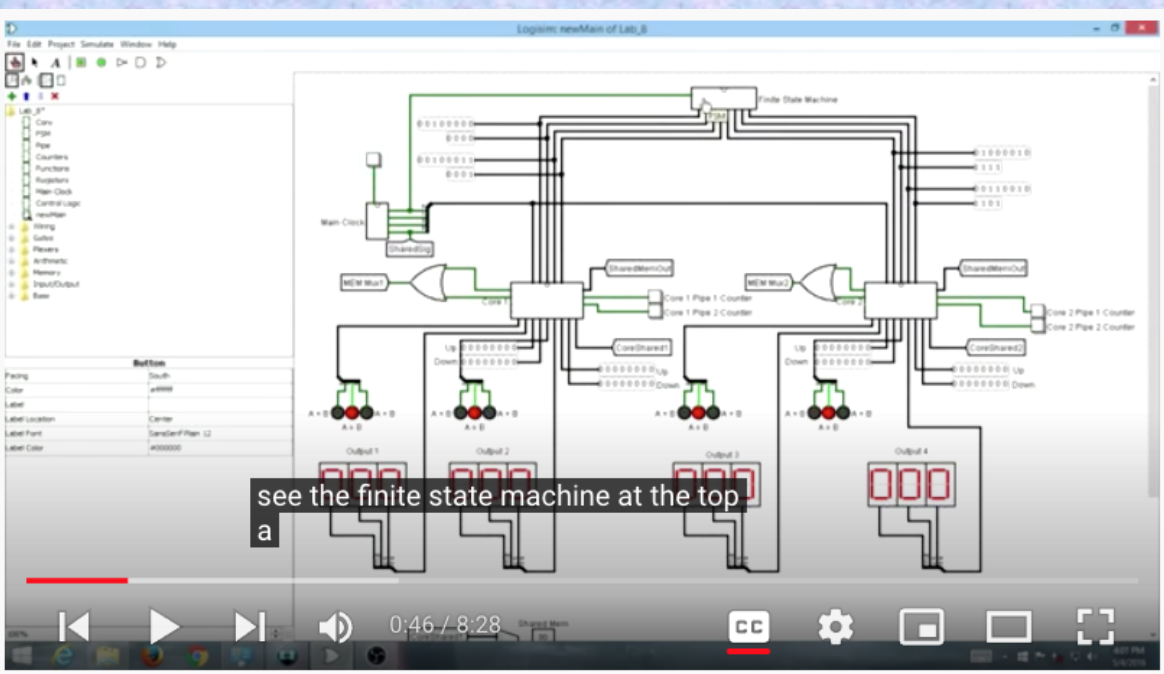


Figure 6. A simple digital-design course laboratory project to introduce the control and flow of data in a microprocessor or microcontroller.

Wunderlich, J.T. (1999). **Focusing on the blurry distinction between microprocessors and microcontrollers.** In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.

J Wunderlich DUAL-CORE SUPERSCALAR Project



2016 EGR/CS 433 Advanced Computer Engineering Final Lab Project Video SUPERSCALAR DUAL CORE

53 views... 0 DISLIKE SHARE DOWNLOAD SAVE ...

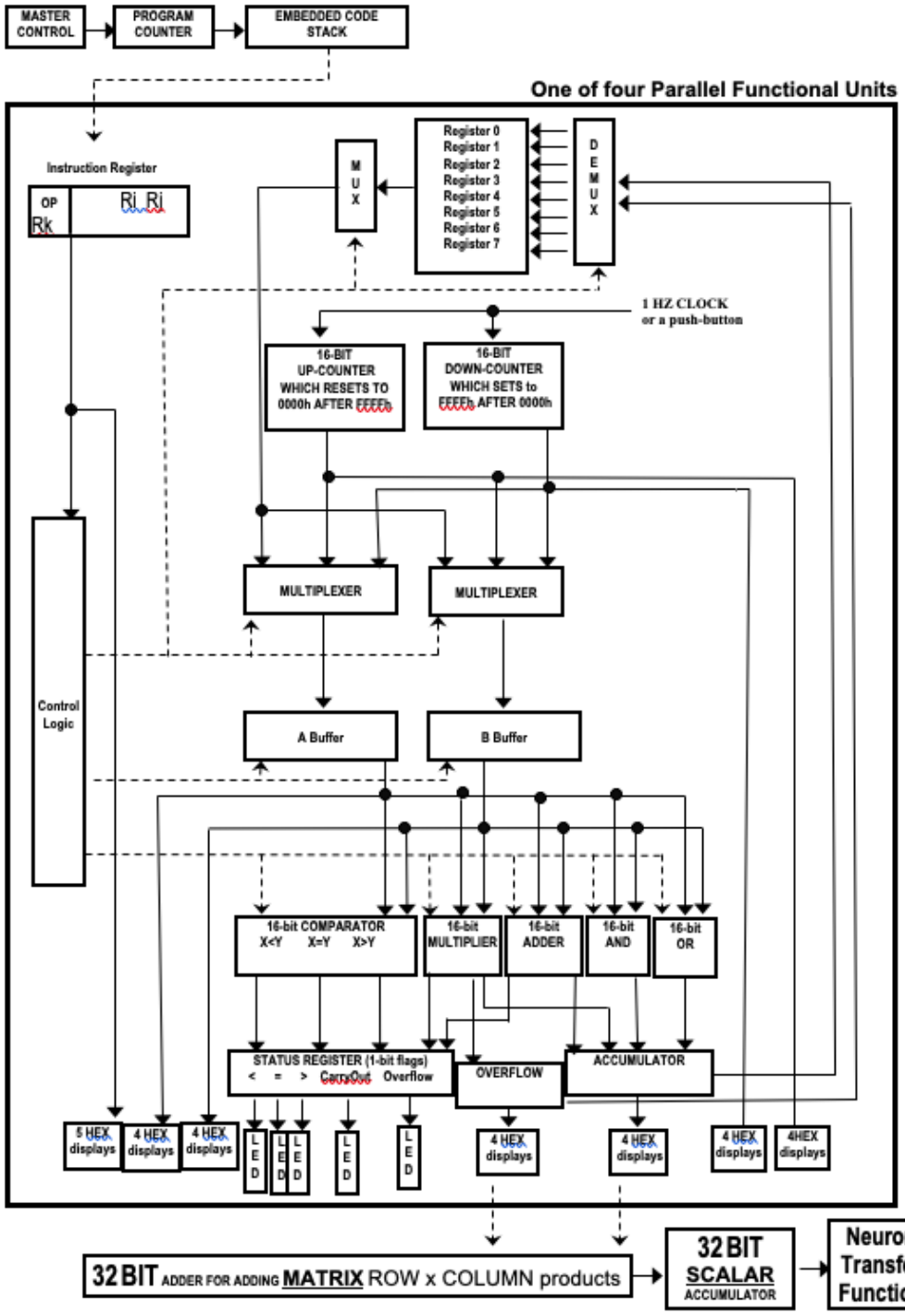
Joseph Wunderlich
29 subscribers

ANALYTICS EDIT VIDEO

"Computer Instruction-Set Design and Implementation with DUAL 2-way SUPERSCALAR CORES, and a SHARED MEMORY"
Course Syllabus: http://users.etown.edu/w/wunderjt/syl...

https://www.youtube.com/watch?v=UgKWvLGx0ZM&list=PLK3MJsXEYEQIGw3tmIkjsBfZ49rr_GAmv&index=11

J Wunderlich VECTOR-REGISTER Project



MACHINE INSTRUCTION SET

SCALAR ARITHMETIC ADDITION

```
00h (OP-CODE = 00000000) Ri + counter#1 →Rk
01h (OP-CODE = 00000001) Ri + counter#2 →Rk
02h (OP-CODE = 00000010) Ri + Rj →Rk
03h (OP-CODE = 00000011) counter#1 + counter#2 →Rk
```

SCALAR ARITHMETIC SUBTRACTION

```
04h to 07h (OP-CODE = 000001XX) Reserved for subtraction instructions
```

SCALAR ARITHMETIC MULTIPLICATION

```
08h (OP-CODE = 00001000) Ri x counter#1 →Rk, overflow →Rk+1 (wrap to R0)
09h (OP-CODE = 00001001) Ri x counter#2 →Rk, overflow →Rk+1 (wrap to R0)
0Ah (OP-CODE = 00001010) Ri x Rj →Rk, overflow →Rk+1 (wrap to R0)
0Bh (OP-CODE = 00001011) counter#1 x counter#2 →Rk, overflow→Rk+1(wrap to R0)
```

SCALAR ARITHMETIC DIVISION

```
0Ch to 0Fh (OP-CODE = 000011XX) Reserved for division instructions
```

SCALAR ARITHMETIC COMPARISON

```
10h (OP-CODE = 00010000) Compare Ri with counter#1 →Rk
11h (OP-CODE = 00010001) Compare Ri with counter#2 →Rk
12h (OP-CODE = 00010010) Compare Ri with Rj →Rk
13h (OP-CODE = 00010011) Compare Counters
```

SCALAR LOGICAL AND

```
20h (OP-CODE = 00100000) Ri AND counter#1 →Rk
21h (OP-CODE = 00100001) Ri AND counter#2 →Rk
22h (OP-CODE = 00100010) Ri AND Rj →Rk
23h (OP-CODE = 00100011) AND counters →Rk
```

SCALAR LOGICAL OR

```
24h (OP-CODE = 00100100) Ri OR counter#1 →Rk
25h (OP-CODE = 00100101) Ri OR counter#2 →Rk
26h (OP-CODE = 00100110) Ri OR Rj →Rk
27h (OP-CODE = 00100111) OR counters →Rk
```

CLEAR

```
30h (OP-CODE = 00110000) Clear Ri
```

MAC (Multiply, Accumulate). Accumulator = Accumulator + (Ri x Rj) *considering overflow also*

```
40h (OP-CODE = 01000000) Step 1: Accumulator →Rk+3 (wrap to R0+)
                         Step 2: Overflow →Rk+4 (wrap to R0+)
                         Step 3: Ri x Rj →Rk, overflow →Rk+1 (wrap to R0)
                         Step 4: (Rk+3)+Rk →Rk
                         Step 5: (Rk+4)+(Rk+1)+Carry →Rk+1
```

VECTOR/ARRAY / MATRIX and NEURON INSTRUCTIONS

Vi, Vj, and Vk are created from Ri's, Rj's, and Rk's of the four parallel functional units

VECTOR ARITHMETIC ADDITION

```
82h (OP-CODE = 10000010) Vi + Vj →Vk
```

VECTOR ARITHMETIC SUBTRACTION

```
84h to 87h (OP-CODE = 100001XX) Reserved for subtraction instructions
```

VECTOR ARITHMETIC MULTIPLICATION

```
0Ah (OP-CODE = 10001010) Vi x Vj →Vk, overflow →Vk+1 (wrap to R0)
```

VECTOR ARITHMETIC DIVISION

```
8Ch to 8Fh (OP-CODE = 100011XX) Reserved for division instructions
```

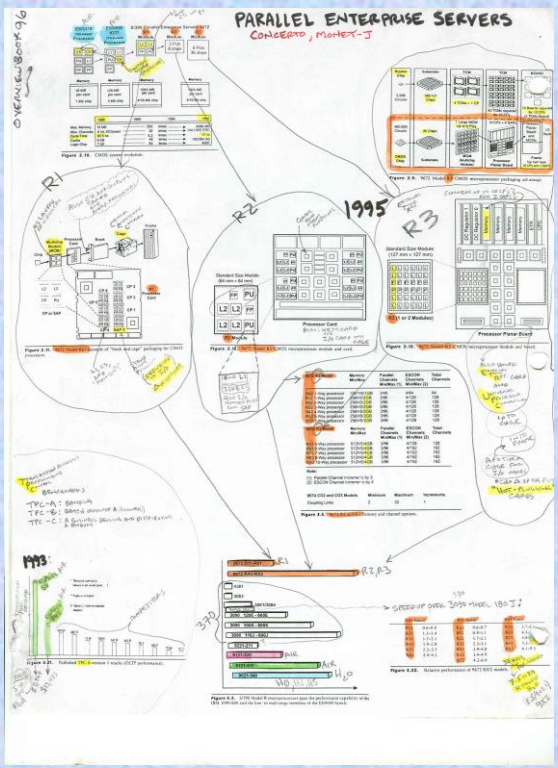
MATRIX ROW x COLUMN (i.e., Dot-Product)

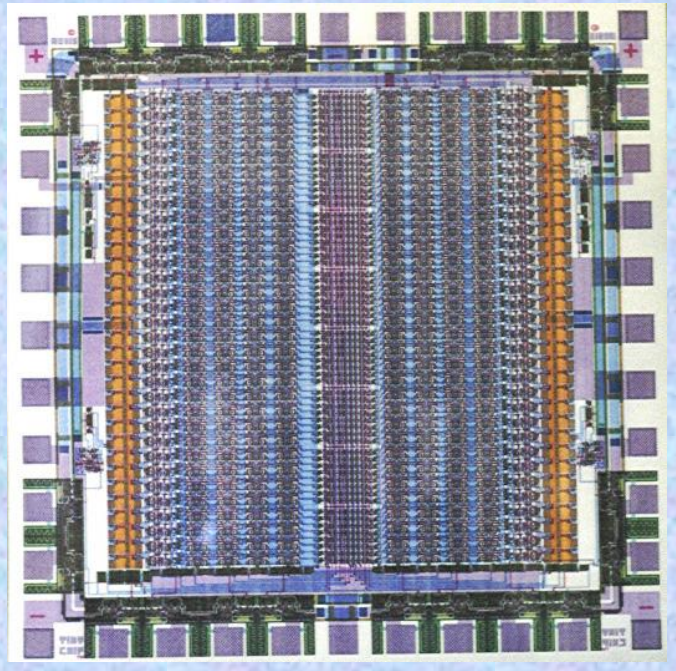
```
C0h (OP-CODE = 11000000) Vi x Vj →Vk, overflow →Vk+1 (wrap to R0)
                         Vk(1)+Vk(2)+Vk(3)+Vk(4) → 32-Bit Scalar Accumulator
```

NEURON TRANSFER FUNCTION

```
E0h (OP-CODE = 11100000) Vi x Vj →Vk, overflow →Vk+1 (wrap to R0)
                         Vk(1)+Vk(2)+Vk(3)+Vk(4) → 32-Bit Scalar Accumulator
                         32-Bit Scalar Accumulator → Neuron Transfer Function
```

More in my intro lecture(s) to EGR/CS433 / EGR430:
"CPU'S, GPU'S, SUPERCOMPUTERS, & NEUROCOMPUTERS, AND MY IBM SUPERCOMPUTER RESEARCH, + PREVIOUS NEURAL NETWORK PROCESSOR DESIGNS" PPTX-w/audio PDF MP4 YouTube





And on my CV:
http://users.etown.edu/w/wunderjt/Wunderlich,JoeCV.pdf

IBM S/390 HARDWARE DEVELOPMENT LAB *(Poughkeepsie, New York, 6/96-7/98)*

Researcher & Hardware Development Engineer (Advisory-Level)

Helped develop Symmetric Multi-Processor (SMP) mainframe-supercomputer architectures (jointly developed with IBM Germany) by engineering systems-level software and part of the SAK (Systems Assurance Kernel) operating system for QUALITY-CONTROL / VERIFICATION to "stress" features and force hardware failures through pseudo-random generation of correlated machine states and operating scenarios. Machines included 20 multicore processors (18 CPU and 2 I/O); divisible into 15 logical partitions and scalable to 512 processors to fit inside a $1M vending-machine size box; Scalable/connectable to other mainframes & supercomputers via a dynamic optical interconnect (IBM Parallel Sysplex). Engineered software to run in three environments: VLSI circuit simulation, prototype hardware test, and manufacturing. New 64-bit processing (address and data) required simulating 64-bit arithmetic and virtual-addressing to test simulated 64-bit prototype architectures using 32-bit machines. Prototypes were released as "IBM eServer zSeries" (now called "IBM Z"). My research included microprocessor branch-prediction verification strategies in a multiprocessor environment; and theory for hardware verification with seven correlated random number generators. My development projects included writing 20,000 lines of high-level language (PL/X) and S/390 assembly code including operating system application interfaces (API's). My RNG API code was also translated into C for IBM AS/400 minicomputers and RS-6000 (AIX type UNIX) workstations (the predecessor of POWER7 supercomputers like "Watson") requiring supervising an engineer in Austin TX via the IBM intranet. Other projects included verification programs for cache coherency, virtual addressing, space-switching, linkage control, and 125 new IEEE floating-point instructions (to supplement IBM Hex floating-point). All 1400 IBM S/390 machine instructions were tested (including vector-register instructions for add-on vector-register unit). A patent process was initiated for my random number theory and API's.