

CS 5803

*Introduction to High Performance Computer
Architecture: Concurrency*



A.R. Hurson
323 CS Building,
Missouri S&T
hurson@mst.edu

Introduction to High Performance Computer Architecture



◆ Outline

- * Multifunctional systems
- * CDC 6600
- * Computation Gap
 - Definition
 - How to reduce computation gap
- * Reducing the computation gap in CDC 6600
- * Concurrency
 - Definition
 - Classifications
 - Parallel systems
 - Pipeline systems
 - Multiprocessor systems

Introduction to High Performance Computer Architecture

Note, this unit will be covered in three weeks. In case you finish it earlier, then you have the following options:

- 1) Take the early test and start CS5803.module7
- 2) Study the supplement module
(supplement CS5803.module6)
- 3) Act as a helper to help other students in studying CS5803.module6

Note, options 2 and 3 have extra credits as noted in course outline.

Enforcement of background

Glossary of prerequisite topics

Familiar with the topics? **No** → Review CS5803.module6.background

Yes

Take Test

Pass? **No** → Remedial action

Yes

Glossary of topics

Familiar with the topics? **No** → Take the Module

Yes

Take Test

Pass? **No** → Take the Module

Yes

Options

Study next module?

Lead a group of students in this module (extra credits)?

Study more advanced related topics (extra credits)?

At the end: take exam, record the score, and impose remedial action if not successful

Extra Curricular activities

◆ Multifunctional Systems

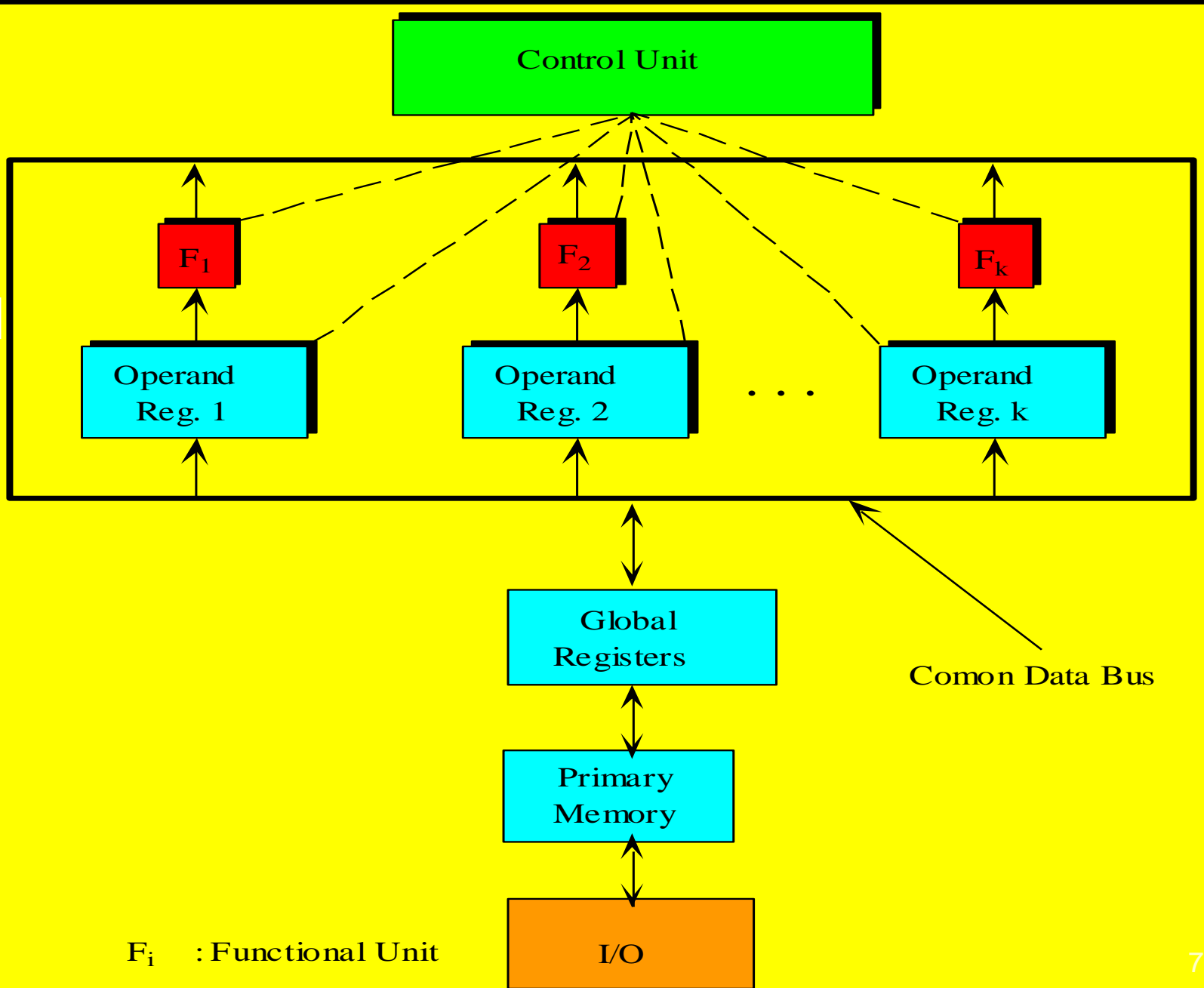
★ General Configuration

- The system is composed of a **single control unit** and a **processor**.
- The processing unit is composed of **several independent functional units**.
- Each functional unit has a set of local (private) registers and all functional units share a set of global registers, which hold intra-functional operands and act as a buffer for the memory unit.
- Each functional box is **tailored** around a specific operation.

◆ Multifunctional Systems

★ General Configuration

- The global registers share a common bus and a switching network which, together, allow fast data transmission from point to point.
- The primary memory should support a high processor bandwidth.
- The control unit is responsible for the resolution of register and functional unit conflicts and scheduling of their operations.
- Functional units work independently in asynchronous mode.



◆ Multifunctional Systems

★ Historical Development

- CDC 6600 1964
- CDC 7600 1969
- CRAY-1 1976
- CRAY X-MP 1982
- CRAY-2 1985
- CRAY Y-MP 1988
- CRAY-3 and CRAY-4 ?
- Superscalar
- VLIW



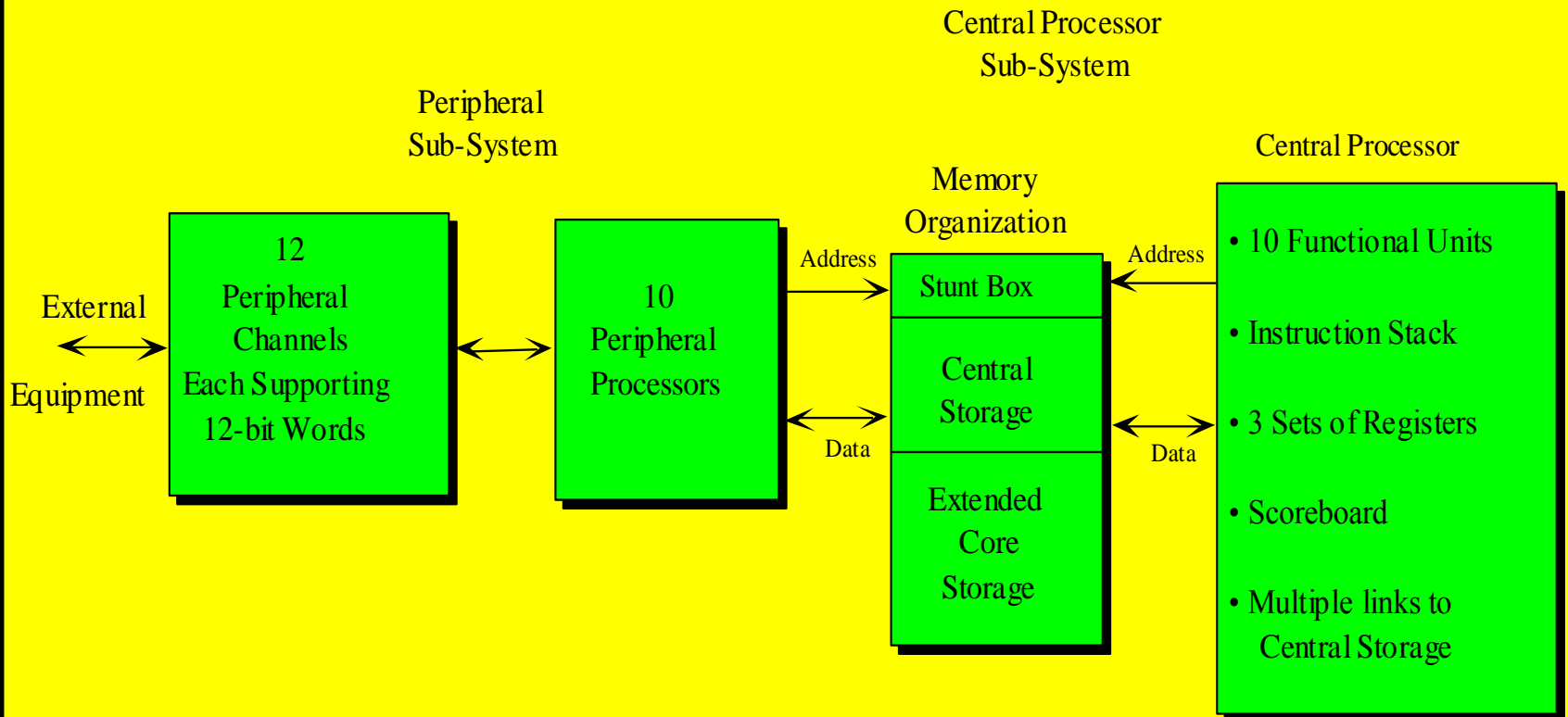
◆ CDC 6600

★ General Philosophy

- Separation of the input/output operations from the central processor operations.
- Highly parallel central processing unit — multifunctional organization.
- Hierarchical and Interleaved memory organization.
- Pipelined instruction cycle.
- Overall estimated performance 4.5 MFLOPS.

Introduction to High Performance Computer Architecture

◆ CDC 6600 — Block Diagram



Introduction to High Performance Computer Architecture

◆ CDC 6600

- ★ Peripheral Sub-System is composed of two parts:
 - Peripheral Channels
 - Peripheral Processing Units

Introduction to High Performance Computer Architecture

◆ CDC 6600

★ Memory Organization

- Central Storage
- Stunt Box
- Extended Core Storage

◆ CDC 6600

★ Central Storage

- A collection of 32 independent banks of 60-bit words.
- Banks are arranged in an interleaved fashion with a memory cycle time of 10 clock cycles.

◆ CDC 6600

★ Stunt Box

- Organization of Stunt Box has already been discussed. If you recall, it is designed to provide a maximum flow of addresses to the central storage. This has been achieved by not allowing accesses to a busy bank to stop other accesses to the central storage.

◆ CDC 6600

★ Extended Core Storage

- Backup storage for central storage.
- A collection of up to 16 banks of 480-bit words.
- Storage cycle time is 32 clock cycles.

◆ CDC 6600

★ Central Processor

- The central processor is based on a high degree of **functional parallelism**.
- Functional parallelism is achieved by the use of many **functional units** and a number of essential supporting modules.

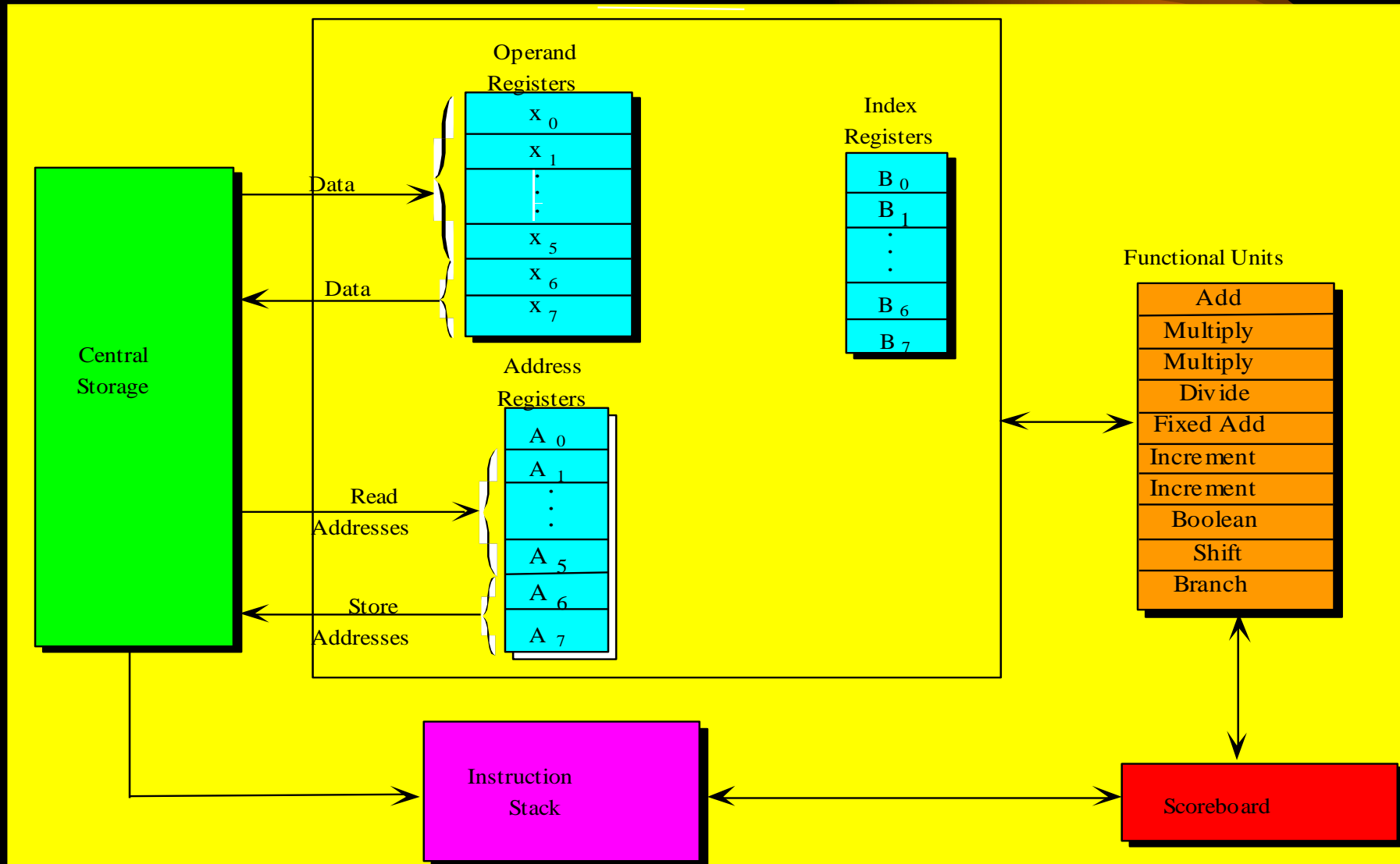
Introduction to High Performance Computer Architecture

◆ CDC 6600

★ Central Processor is composed of:

- 24 Registers
- Instruction Stack
- Functional Units
- Scoreboard

Introduction to High Performance Computer Architecture



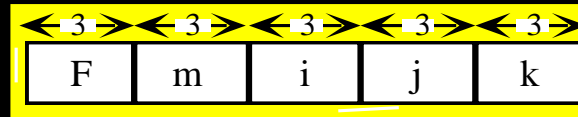
Introduction to High Performance Computer Architecture

◆ CDC 6600

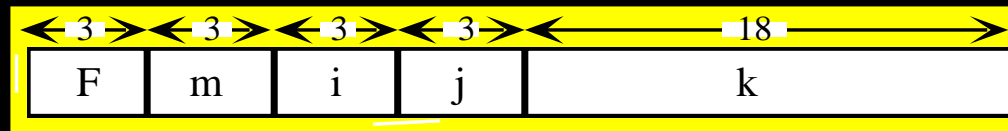
★ Central Processor

- Two formats of instructions are supported:

- Short format instruction - 15 bits.



- Long Format instruction - 30 bits.



◆ CDC 6600

★ Central Processor

- F, m fields collectively represent the op-code.
- i, j, k fields refer to one of the A, X, B registers.
- k field represents an immediate value used as a constant or branch destination.

◆ CDC 6600

★ 24 Registers

- Eight operand registers (X-registers) - 60 bits.
- Eight address registers (A-registers) - 18 bits.
- Eight index registers (B-registers) - 18 bits.
- Operand registers are paired up one-for-one with a corresponding address register.
- Five address-operand register pairs are used for read and two are used for write.

Introduction to High Performance Computer Architecture

◆ CDC 6600

★ Instruction Stack

- Instruction Stack is a collection of eight 60-bit registers — instruction register and seven buffer registers. This configuration allows up to 32 previously fetched instructions to be readily available in the instruction stack.
- A reference to an instruction resident in the instruction stack is granted by the instruction stack. This:
 - Allows **faster access** to the referenced instruction, and
 - Reduces the main **memory contention**.

◆ CDC 6600

★ Functional Units

- Ten independent functional units make up the arithmetic and logic portion of the CDC 6600.

These units **may operate simultaneously**:

- Floating Point Add
- Floating Point Multiply (2)
- Floating Point Divide
- Fixed Point Add
- Increment (2)
- Boolean
- Shift
- Branch

Unit	Execution Time (clock cycles)	Semantics
Floating Point Multiply	10	$X_i \leftarrow X_j * X_k$
Floating Point Add/Sub	4	$X_i \leftarrow X_j + X_k$
Fixed Point Add/Sub	3	$X_i \leftarrow X_j + X_k$
Increment/Decrement	3	18 bits operand, indexing, conditional branch, reading, storing
Boolean	3	60 bits logic operation
Shift	3-4*	Shift, normalize,* pack, unpack, mask
Floating Point divide	29	$X_i \leftarrow X_j / X_k$
Branch	$8^1, 9^2, 14^3$ if next inst. in stack else add + 6	

Each two consecutive words are fetched after 8 minor cycles (if no memory conflict exists).

1 if increment unit is used
2 if floating point add is used
3 if constant + contents of B register is used

◆ CDC 6600

★ Scoreboard

- Scoreboard is a hardware unit which keeps track of instruction conflicts in an attempt to **issue independent instructions** as fast as possible.
- Scoreboard holds the current status of each instruction that is currently being executed or waiting to be executed in the functional units.

◆ CDC 6600

★ Instruction Fetch

- An instruction word is fetched into the instruction register. Instruction words are made up of **four parcels** of 15 bits each.
- During each clock cycle the contents of two successive parcels (**instruction packet**) are fetched and passed through a series of registers - U_0 , U_1 , U_2 (instruction pipe). Two consecutive instruction packets overlap 15 bits of information.

◆ CDC 6600

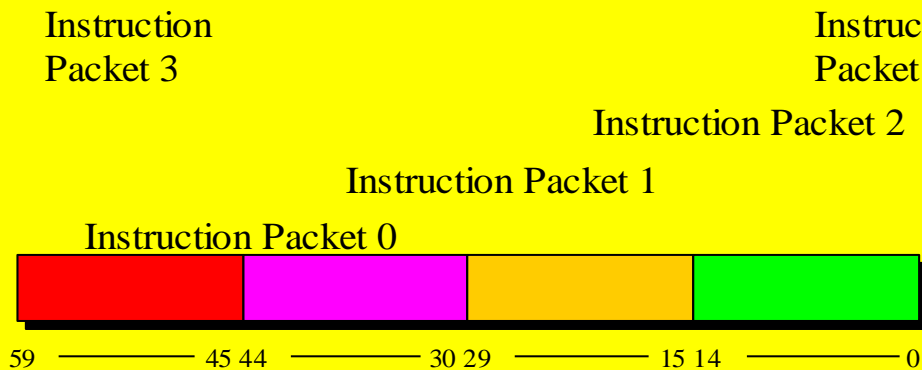
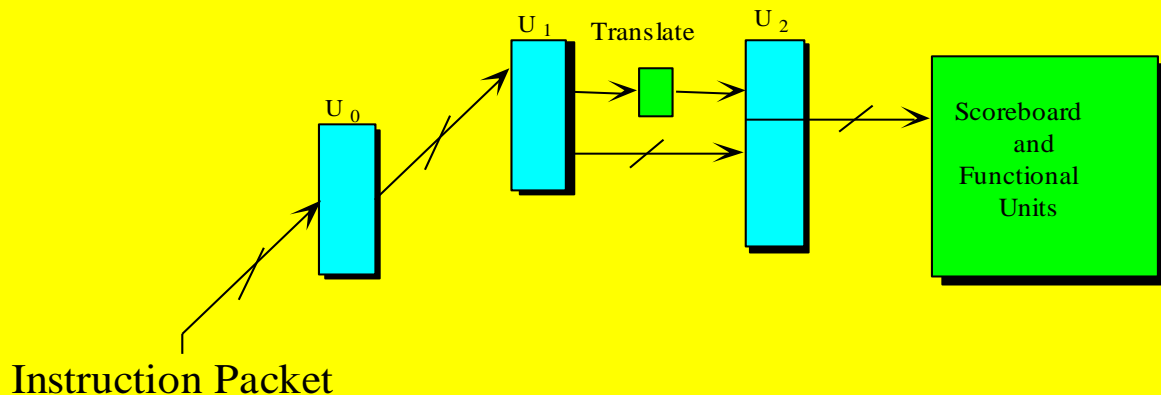
★ Instruction Pipe

- During the transition from U_1 to U_2 the instruction packet will be **translated**, this determines the length of the instruction, the functional unit being requested and the operand register(s) involved.
- If **no conflict** exists, the instruction is **issued** to the appropriate functional unit. Otherwise, the instruction may not be issued, halting (temporarily) the flow of the information in the **instruction pipe**.

Introduction to High Performance Computer Architecture

◆ CDC 6600

★ Instruction Pipe



◆ CDC 6600

★ Instruction Pipe

- Note after each long instruction, the contents of the next instruction packet is **an invalid instruction** and will be skipped.

◆ CDC 6600

★ Conflicts and Their Resolutions

- In a parallel environment, simultaneous execution of instructions is possible as long as instructions are **independent** of each other.
- In the case of dependence, the degree of dependence should be recognized and proper resolution should be taken.

◆ CDC 6600 — Conflicts and Their Resolutions

★ First Order Conflict

- Functional Unit Conflict (conflict over hardware resources)

$$X_6 = X_1 + X_2$$

●

●

●

$$X_5 = X_3 + X_4$$

◆ CDC 6600 — Conflicts and Their Resolutions

★ First Order Conflict

- Result Register Conflict (Write After Write)

$$X_6 = X_1 * X_2$$

•
•
•

$$X_6 = X_4 + X_5$$



◆ CDC 6600 — Conflicts and Their Resolutions

★ First Order Conflict

- Resolution: The second instruction is not issued until the first instruction is completed (instruction pipe is temporarily halted).

◆ CDC 6600 — Conflicts and Their Resolutions

★ Second Order Conflict (Read After Write)

- This conflict occurs when an instruction requires the result of a previously issued, and as yet uncompleted, instruction as an input operand.

$$X_6 = X_1 * X_2$$

•

•

•

$$X_7 = X_5 / X_6$$



◆ CDC 6600 — Conflicts and Their Resolutions

★ Second Order Conflict (Read After Write)

- Resolution: Functional unit is reserved and instruction is issued, but its execution is delayed.



◆ CDC 6600 — Conflicts and Their Resolutions

★ **Third Order Conflict (Write After Read)**

- This conflict occurs when an instruction is called on to store its result in a register which is to be used as an input operand for a previously issued, but as yet not started, instruction.

◆ CDC 6600 — Conflicts and Their Resolutions

★ Third Order Conflict (Write After Read)

$$X_3 = X_1 / X_2$$

•

•

•

$$X_5 = X_4 * X_3$$

•

•

•

$$X_4 = X_0 + X_6$$



◆ CDC 6600 — Conflicts and Their Resolutions

★ **Third Order Conflict (Write After Read)**

- **Resolution:** The instruction is issued, its execution will be started, but its result will be held in the functional unit as long as the conflict exists.

◆ CDC 6600 — Example

- ★ Within the scope of CDC6600, calculate the timing chart for the following expression:

$$Y = AX^2 + BX + C$$

Introduction to High Performance Computer Architecture

◆ CDC 6600 — Example

Word #	Instruction	Semantics	I S S U E	S T A R T	R E S U L T	U N I T H	F E T C H	S T O R E
N ₁	A ₁ =A ₁ +K ₁	FETCH X	1	1	4	5	9	
	A ₂ =A ₂ +K ₂	FETCH A	3	3	6	7	11	
N ₂	X ₀ =X ₁ *X ₁	FORM X ²	9	9	19	20		
	X ₆ =X ₀ *X ₂	FORM AX ²	10	19	29	30		
	A ₃ =A ₃ +K ₃	FETCH B	11	11	14	15	19	
N ₃	A ₄ =A ₄ +K ₄	FETCH C	17	17	20	21	25	
	X ₃ =X ₃ *X ₁	FORM BX	20	20	30	31		
	X ₅ =X ₆ +X ₃	FORM A X ² +BX	21	30	34	35		
N ₄	X ₇ =X ₅ +X ₄	FORM Y	35	35	39	40		
	A ₇ =A ₇ +K ₅	STORE Y	36	39	42	43		47

◆ Computation Gap

- ★ **Computation gap** is defined as the difference between computational power demanded by the application environments and computational capability of the existing computers.
- ★ Today, one can find many applications which require orders of magnitude more computations than the capability of the most powerful computers.

◆ Computation Gap

- ★ Computational requirements of some applications
 - It is estimated that the so called **Problem Solving and Inference Systems** require an environment with the computational power in the order of 100 MLIPS to 1 GLIPS (1 LIPS \approx 100-1000 instructions).

◆ Computation Gap

- ★ Computational requirements of some applications
 - Experiences in **Fluid Dynamics** have shown that the conventional super-computers can calculate **steady 2-dimensional flow** in minutes. However, conventional super-computers require up to 20 hours to handle **time dependent 2-dimensional flow** or **steady 3-dimensional flows** on simple objects.



◆ Computation Gap

- ★ Computational requirements of some applications
 - Numerical Aerodynamics Simulator requires an environment with a sustained speed of 1 billion FLOPS.
 - Strategic Defense Initiative requires a distributed, fault tolerant computing environment with a processing rate of 600 MOPS.

◆ Computation Gap

- ★ Computational requirements of some applications
 - U.S. Patent Office and Trademark has a database of size 25 terabytes subject to search and update.
 - An angiogram department of a mid-size hospital generates more than $64 * 10^{11}$ bits of data a year.
 - NASA's Earth Observing System will generate more than 11,000 terabytes of data during the 15-year time-period of the project.

◆ Computation Gap

★ Computational requirements of some applications

- It was estimated that in 2002, 5 exabytes (1 exabyte= 10^{18} bytes) which is approximately equal to all words spoken by human beings) of new information was generated.
- The TREC database holds around 800 million static pages having 6 trillion bytes of plain text equal to the size of a million books.
- The Google system routinely accumulates millions of pages of new text information every week.

◆ Computation Gap

★ Performance of some computers

● CDC STAR-100	25-100	MFLOPS
● DAP	100	MFLOPS
● ILLIAC IV	160	MFLOPS
● HEP	160	MFLOPS
● CRAY-1	25-80	MFLOPS
● CRAY X-MP(1)	210	MFLOPS
● CRAY X-MP(4)	840	MFLOPS

◆ Computation Gap

★ Performance of some computers

● CRAY-2	250	MFLOPS
● CDC CYBER 200	400	MFLOPS
● HitachiS-810(10)	315	MFLOPS
● HitachiS-810(20)	630	MFLOPS
● FujitsuFACOM VP-50	140	MFLOPS
● FujitsuFACOM VP-100	285	MFLOPS
● FujitsuFACOM VP-200	570	MFLOPS
● FujitsuFACOM VP-400	1,140	MFLOPS

◆ Computation Gap

★ Performance of some computers

● NEC SX-1	570	MFLOPS
● NEC SX-2	1,300	MFLOPS
● IBM RP3	1,000	MFLOPS
● MPP 8-bit integer	1545-6553	MIPS
● MPP 12-bit integer	795-4428	MIPS
● MPP 16-bit integer	484-3343	MIPS
● MPP 32-bit FL	165-470	MIPS
● MPP 40-bit FL	126-383	MIPS



◆ Computation Gap

★ Performance of some computers

- NEC (Earth System) 35 tera FLOPS
- IBM Blue Gene 70 tera FLOPS

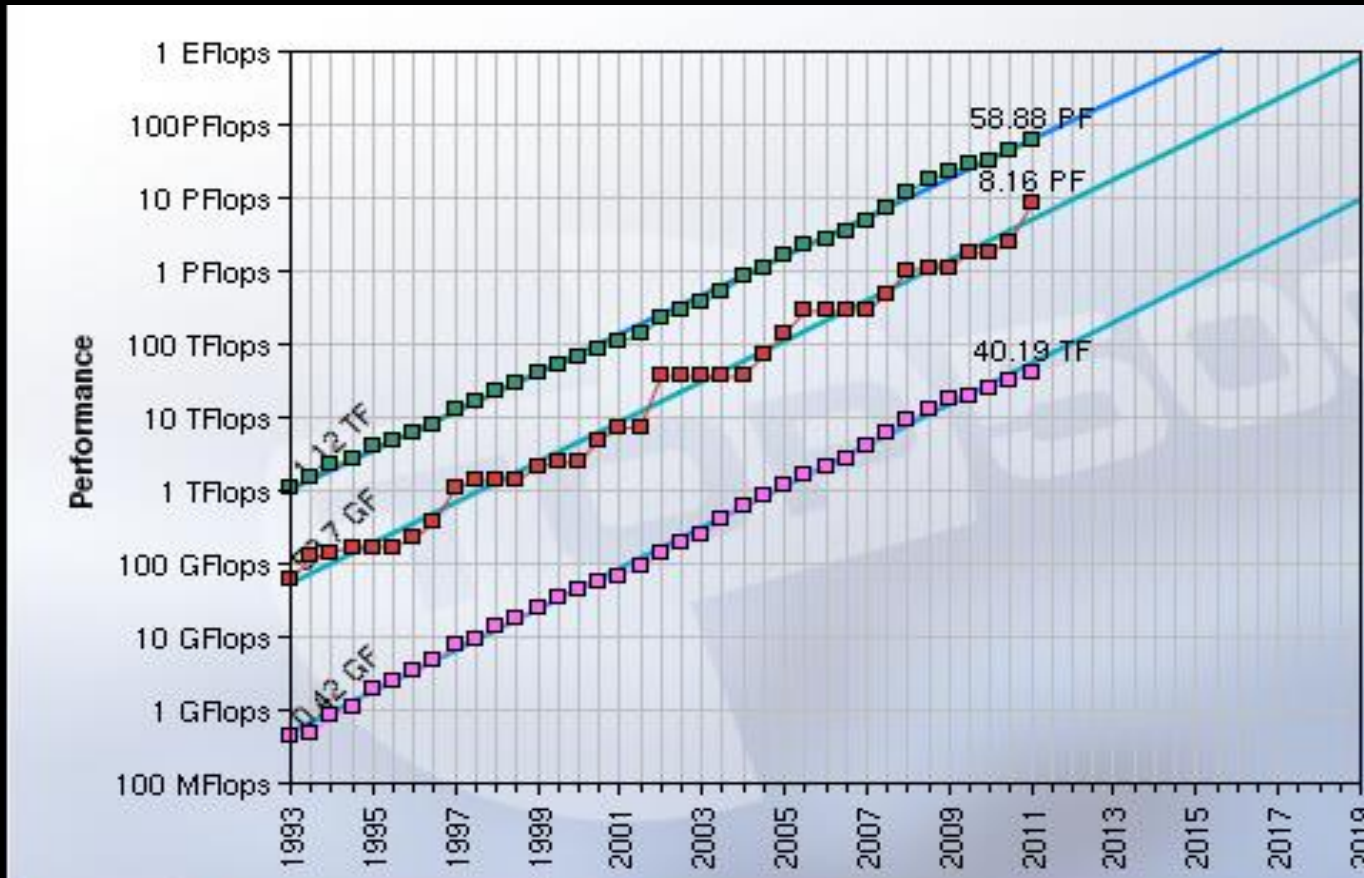
Introduction to High Performance Computer Architecture

The Top10 supercomputers

Rank	Site	Computer/Year Vendor	Country	Cores	R _{max} (Pflops)	R _{peak} (Pflops)	Power (MW)
1	RIKEN Advanced Institute for Computational Science	K computer, SPARC64 VIIIfx 2.0GHz,/ 2011 Fujitsu	Japan	548,352	8.162	8.774	9.899
2	National Supercomputing Center in Tianjin	Tianhe-1A - NUDT / 2010 NUDT	China	186,368	2.566	4.701	4.040
3	DOE/SC/Oak Ridge National Laboratory	Jaguar - Cray XT5-2.6 GHz / 2009 Cray Inc.	USA	224,162	1.759	2.331	6.951
4	National Supercomputing Centre in Shenzhen	Nebulae - Dawning TC3600 Blade/ 2010 Dawning	China	120,640	1.271	2.984	2.580
5	GSIC Center, Tokyo Institute of Technology	TSUBAME 2.0/2010 NEC/HP	Japan	73,278	1.192	2.288	1.399
6	DOE/NNSA/LANL/SNL	Cielo - Cray XE6 8-core 2.4 GHz /2011Cray Inc.	USA	142,272	1.110	1.365	3.980
7	NASA/Ames Research Center/NAS	Pleiades - 2.93 Ghz,/ 2011 SGI	USA	111,104	1.088	1.315	4.102
8	DOE/SC/LBNL/NERSC	Hopper - Cray XE6 12-core 2.1 GHz / 2010 Cray Inc.	USA	153,408	1.054	1.289	2.910
9	Commissariat a l'Energie Atomique (CEA)	Tera-100 - Bull bullx super-node S6010/S6030 / 2010 Bull SA	France	138,368	1.050	1.255	4.590
10	DOE/NNSA/LANL	Roadrunner - 3.2 Ghz /2009 IBM	USA	122,400	1.042	1.376	2.346

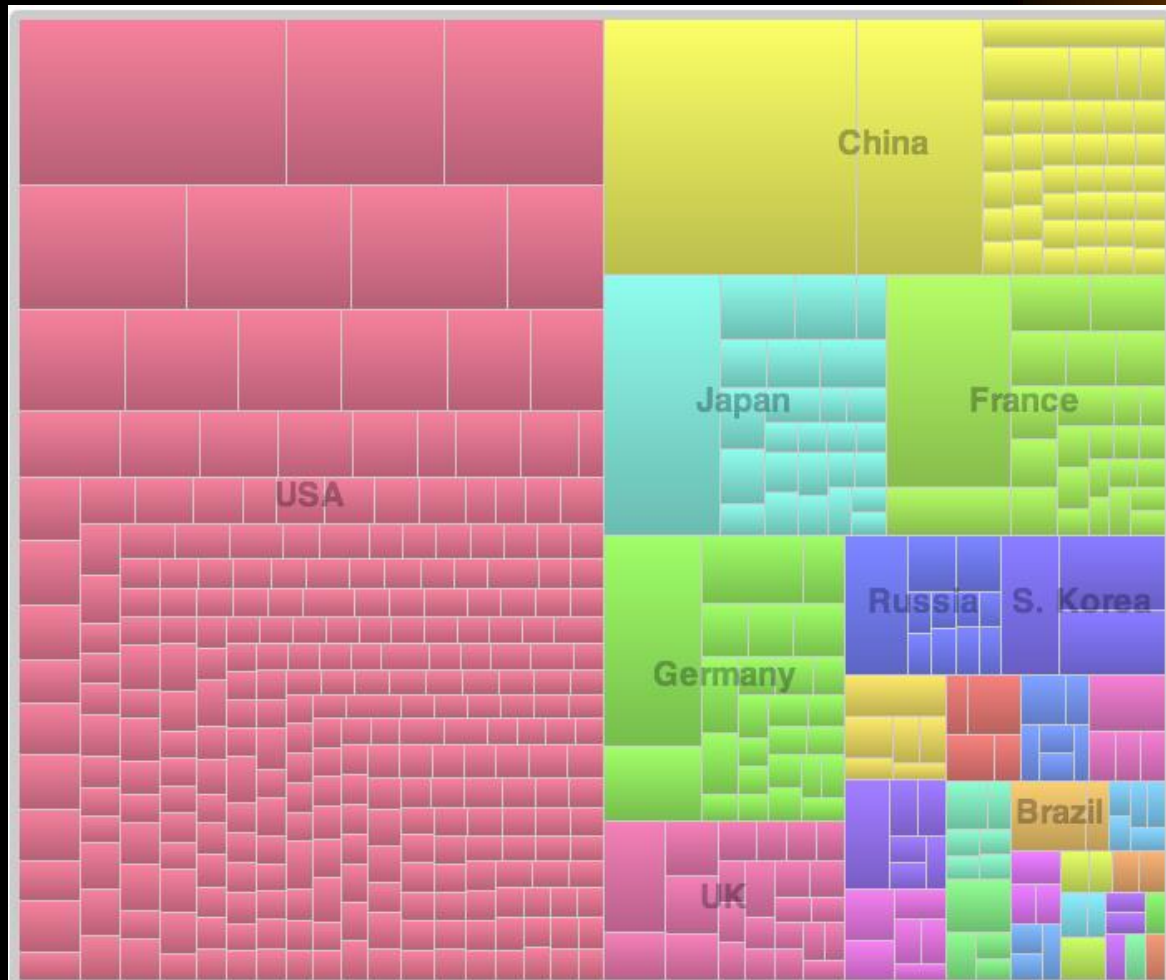
Introduction to High Performance Computer Architecture

Trend in Supercomputer technology



- #1
- Sum
- #500

Countries Share



Absolute Counts

US: 274

China: 41

Germany: 26

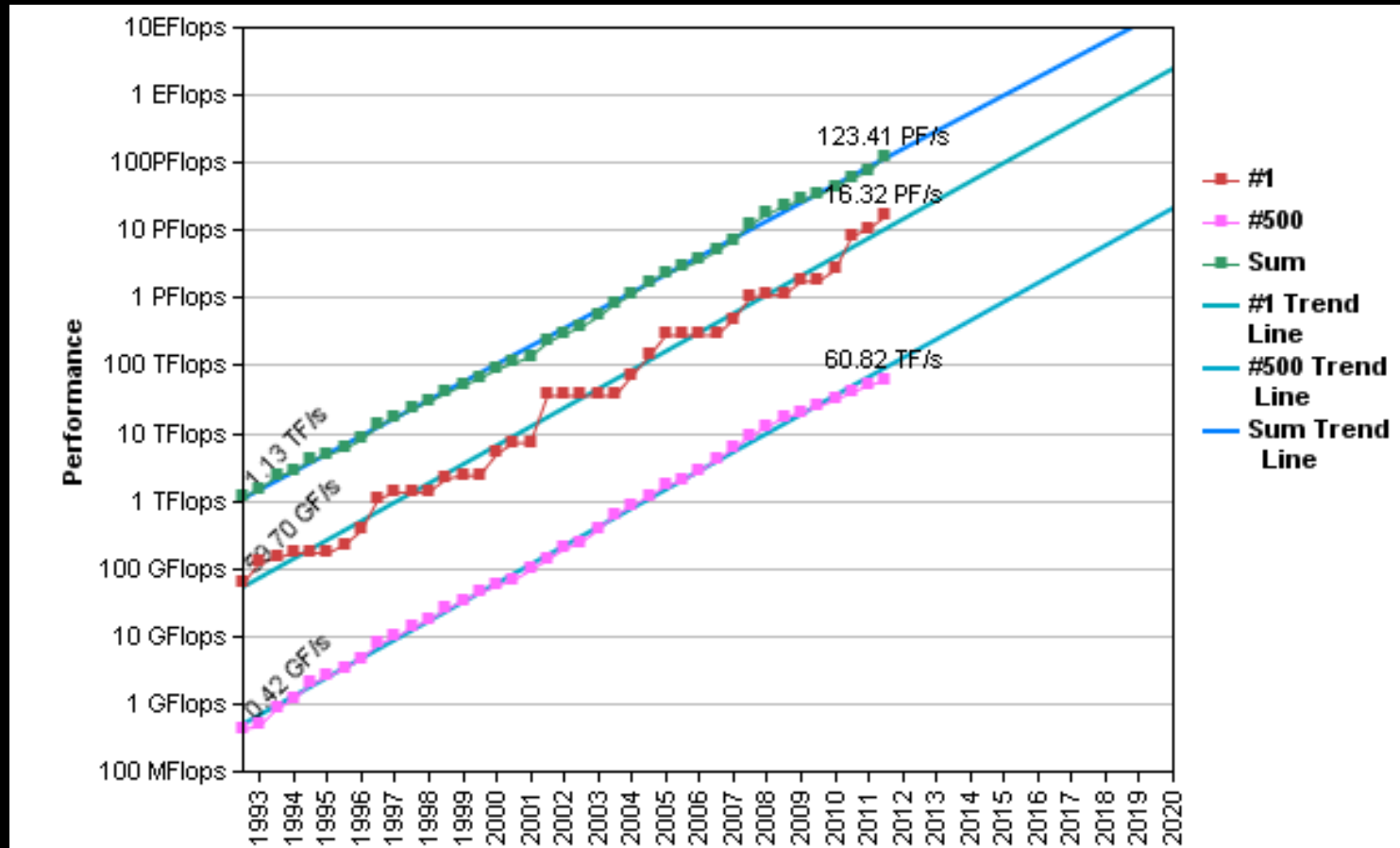
Japan: 26

France: 26

UK: 25

Introduction to High Performance Computer Architecture

Trend in Supercomputer technology (June 2012)



Rapid change in Countries Share

Countries	Count	Share %	Rmax Sum (GF)	Rpeak Sum (GF)	Processor Sum
Australia	6	1.20 %	400406	552142	40344
Austria	2	0.40 %	188670	243386	26172
Belgium	2	0.40 %	83840	151472	16704
Brazil	2	0.40 %	269730	330445	37184
Canada	8	1.60 %	640129	890598	82684
China	61	12.20 %	7136315	14331013	881832
Denmark	2	0.40 %	198408	260395	22218
Finland	2	0.40 %	117858	180690	18640
France	25	5.00 %	3180744	4100571	454928
Germany	30	6.00 %	3242111	4181323	568952
India	2	0.40 %	187910	242995	18128
Ireland	1	0.20 %	40495	76608	7200
Israel	2	0.40 %	135361	280436	23928
Italy	5	1.00 %	471746	748248	42080
Japan	26	5.20 %	11182236	13641290	832838
Korea, South	4	0.80 %	950833	1126280	123384
Netherlands	1	0.20 %	50924	64973	3456
Norway	1	0.20 %	40590	51060	5550
Poland	5	1.00 %	315075	448204	44274
Russia	12	2.40 %	1341586	2290994	115120
Saudi Arabia	4	0.80 %	359240	414841	81920
Singapore	2	0.40 %	94073	144562	13192
Spain	2	0.40 %	135860	197696	14160
Sweden	5	1.00 %	489530	661642	75280
Switzerland	4	0.80 %	317895	383373	49480
Taiwan	2	0.40 %	220504	313570	32148
United Kingdom	27	5.40 %	1872107	2806546	260572
United States	255	51.00 %	25265849	36064596	58037556

Introduction to High Performance Computer Architecture

- ◆ As of June 2013, Tianhe-2 (Milky Way-2) (will be deployed at the National Supercomputer Center in Guangzho, China, by the end of the year) is the fastest super computer with theoretical peak performance of 54.9 PF/s (Linpack Performance of 33.8 PF/s). It is composed of 3,120,000 cores, consuming 17,808 kW.

Introduction to High Performance Computer Architecture

- ◆ Titan, a Cray XK7 system installed at the U.S. Department of Energy's (DOE) Oak Ridge National Laboratory and previously the No. 1 system, is now ranked No. 2. Titan achieved 17.59 petaflop/s on the Linpack benchmark using 261,632 of its NVIDIA K20x accelerator cores. Titan is one of the most energy efficient systems on the list, consuming a total of 8.21 MW and delivering 2,143 Mflops/W.

Introduction to High Performance Computer Architecture

- ◆ Sequoia, an IBM BlueGene/Q system installed at DOE's Lawrence Livermore National Laboratory, also dropped one position and is now the No. 3 system. Sequoia was first delivered in 2011 and has achieved 17.17 petaflop/s on the Linpack benchmark using 1,572,864 cores. Sequoia is also one of the most energy efficient systems on the list, consuming a total of 7.84 MW and delivering 2,031.6 Mflops/W.

◆ Computation Gap


★ Problem


- Suppose a machine capable of handling 10^6 characters per second is in hand. How long does it take to search 25 terabytes of data?

$$\frac{25 * 10^{12}}{10^6} = 25 * 10^6 \text{ sec.} \approx 4 * 10^5 \text{ min.} \approx 7 * 10^3 \text{ Hours} \approx 290 \text{ days}$$

★ NOT PRACTICAL!

★ WHAT ARE THE SOLUTIONS?


- 
- ◆ Computation Gap — How to reduce it?
 - ★ Reduce the amount of needed computations.
 - ★ Improve the speed of the computers:
 - Physical Speed
 - Logical Speed



◆ Computation Gap — Fewer Computation

★ Advances in Software Technology and Algorithms


- Since the early days of computers, the development of software support to maximize hardware utility has stimulated much research.
- Software systems were developed to tailor the embedded hardware features of a system to a specific application.




◆ Computation Gap — Fewer Computation

★ Advances in Software Technology and Algorithms

- Various **data structure** techniques can be used in order to achieve a higher performance.
- Different **algorithms** can be developed to improve performance.

- 
- ◆ Computation Gap — Fewer Computation
 - ★ Advances in Software Technology and Algorithms
 - A **compiler** equipped with an **optimizer** routine improves the performance during the runtime by creating an efficient target language program.



◆ Computation Gap — Fewer Computation

★ Advances in Software Technology and Algorithms

- A **vectorized** and **parallelized compiler** can enhance the performance by detecting the parallelism in an application program and rearranging the instructions in the object program to allow the simultaneous execution of independent instructions or block of instructions on the target machine, during the run time.



◆ Computation Gap — Physical Speed

★ Advances in Technology

- Transition from vacuum tubes to VLSI has made it possible to reduce the gate switching delay and size, and to increase the reliability of the hardware components.



◆ Computation Gap — Physical Speed

★ Advances in Technology

- Within the period of 1940-1980 the processor speed has been increased by more than four orders of magnitude, and logic circuit size and memory cell size have been reduced by factors of 500 and 6400, respectively.



◆ Computation Gap — Physical Speed

★ Advances in Technology

- In 1944, a basic operation was executed in 333 msec.
- About 8 years later, in 1950s, due to the advances in technology, the same basic operation was executed in 282 μ sec.
- In the early 1960s, again, because of the advances in technology, it took 300 nsec to perform the same operation.



◆ Computation Gap — Physical Speed

★ Advances in Technology

- Is it possible to handle the same basic operation in 300 pico sec?
- For the period of 40s-60s, performance (speed) improvement due to the advances in technology has been at the rate of 10^3 per decade. Should we expect the same improvement rate forever?



◆ Computation Gap — Physical Speed

★ Advances in Technology

- In 1970, Intel introduced the first single chip microprocessor, the Intel 4004.
- It had 2,600 manually placed transistors with clocking frequency of 100 KHz.
- Intel 4004 packed as much computing power as the ENIAC.



◆ Computation Gap — Physical Speed

★ Advances in Technology

● Limitations

- Speed of Light and
- Distance

- Light travels $12 * 10^9$ inch per sec. = 12 inch per nsec.
- In 300 pico sec. light travels 4 inches. Therefore, in a hardware unit (basic operation), if the total signal propagation distance is more than 4 inches then it is impossible to execute the same basic operation in 300 pico seconds.

Introduction to High Performance Computer Architecture

Year	1947	1950	1961	1966	1971	1980	1985	1990
Technology	Invention of the Transistor	Discrete Components	SSI	MSI	LSI	VLSI	ULSI*	GSI**
Approximate number of transistors per chip in commercial products	1	1	10	100-1000	1000-20,000	20,000-500,000	>500,000	>1,000,000
Typical Products	-	Junction Transistor and Diode	Planar Devices, Logic Gates, Flip-Flops	Counters, Multiplexers, Adders	8-bit Microproc. ROM, RAM	16 & 32-bit Microproc. Sophisticated Peripherals	Special Processors Real Time Image Proc.	?

* Ultra Large Scale Integration.

** Giant Scale Integration



◆ Computation Gap — Physical Speed

★ Advances in Technology

- Advances in technology reduce the circuit switching delay and miniaturize the hardware circuits. Nevertheless, it cannot transfer signals faster than the speed of light, and cannot eliminate the distance.



◆ Computation Gap — Physical Speed

★ Advances in Technology

- In the late 1960s, Moore predicted that component density on a chip was quadrupling every three or four years. However, as advances in technology approach the limit, the Moore's law is no longer applicable.



◆ Computation Gap — Logical Speed

- ★ One can take two general approaches to improve the logical speed of the system:
 - Architectural advances of the traditional uni-processor systems,
 - Concurrency



◆ Computation Gap — Logical Speed

★ Architectural advances of uni-processor system

- In this case, one has to look at the existing bottlenecks in a uni-processor systems and make an attempt to reduce these bottlenecks.
- As we discuss before access gap is one of the bottlenecks of the traditional uni-processor systems.



◆ Computation Gap — Logical Speed

★ Architectural advances of uni-processor system

- Access gap is defined as the time difference between the CPU cycle time and the main memory cycle time.

◆ Computation Gap — Logical Speed

★ Architectural advances of uni-processor system

- Access gap problem was created by the advances in technology. In fact, in early computers such IBM 704, CPU and main memory cycle time were identical - i.e., 12 μ sec.
- IBM 360/195 had the logic delay of 5 η sec per stage, the CPU cycle time of 54 η sec and the main memory cycle time of .756 μ sec.
- CDC 7600 had the CPU and main memory cycle time of 27.5 η sec and .275 μ sec, respectively.



◆ Computation Gap — Logical Speed

★ Concurrency

- To overcome the technological limitations, computer designers have long been attracted to techniques that are classified under the term of **Concurrency**.



◆ Computation Gap — Logical Speed

★ Concurrency

- Concurrency is a generic term which defines the ability of the computer hardware to simultaneously execute many actions at any instant.
- Within this general term are several well recognized techniques such as **Parallelism**, **Pipelining**, and **Multiprocessing**.

◆ Computation Gap — Logical Speed

★ Concurrency

- These techniques have the same origin and are often hard to distinguish, in practice they are different in their general approach.
- In **parallelism** concurrency is achieved by **replicating** the hardware structure many times, while **pipelining** takes the approach of **splitting** the function to be performed into smaller pieces and allocating separate hardware to each piece.

◆ Computation Gap — Logical Speed

★ Concurrency

- To develop an understanding about concurrency, let us look at several architectural models. As a reminder, an **instruction cycle** is composed of these phases:

- **I-fetch**: to fetch the next instruction and update the program counter.
- **I-decode**: to decode the instruction and fetch operand(s).
- **I-execute**: to execute the instruction.

★ Assume that $T_{I\text{-fetch}} = T_{I\text{-decode}} = T_{I\text{-execute}} = t$

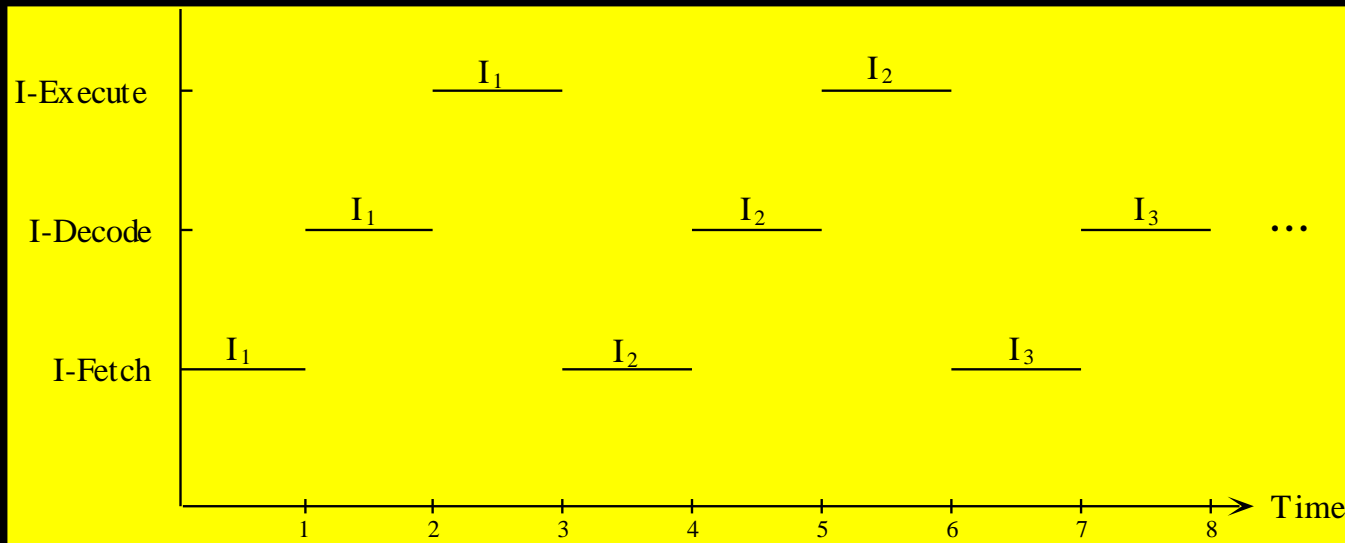


◆ Computation Gap — Logical Speed

★ Concurrency — Model-1

- This architectural model represents a traditional uni-processor organization.
- For each instruction, system goes into an instruction cycle, one cycle at a time.

◆ Computation Gap — Logical Speed
★ Concurrency — Model-1



● Time to execute n instructions

$$T_n = n * (3t)$$



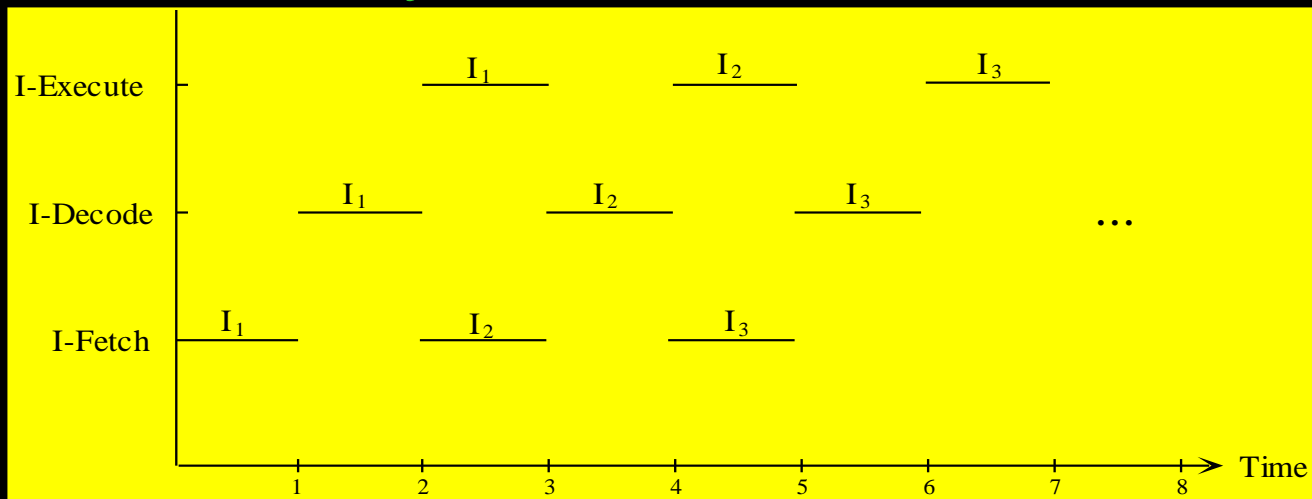
◆ Computation Gap — Logical Speed

★ Concurrency — Model-2

- In this model, I-Fetch phase is **overlapped** with I-Execute phase. This can be easily achieved since, these two phases do not require any common hardware resources.

◆ Computation Gap — Logical Speed

★ Concurrency — Model-2



● Time to execute n instructions

$$T_n = (n-1) * 2t + 3t = (2n + 1)t$$

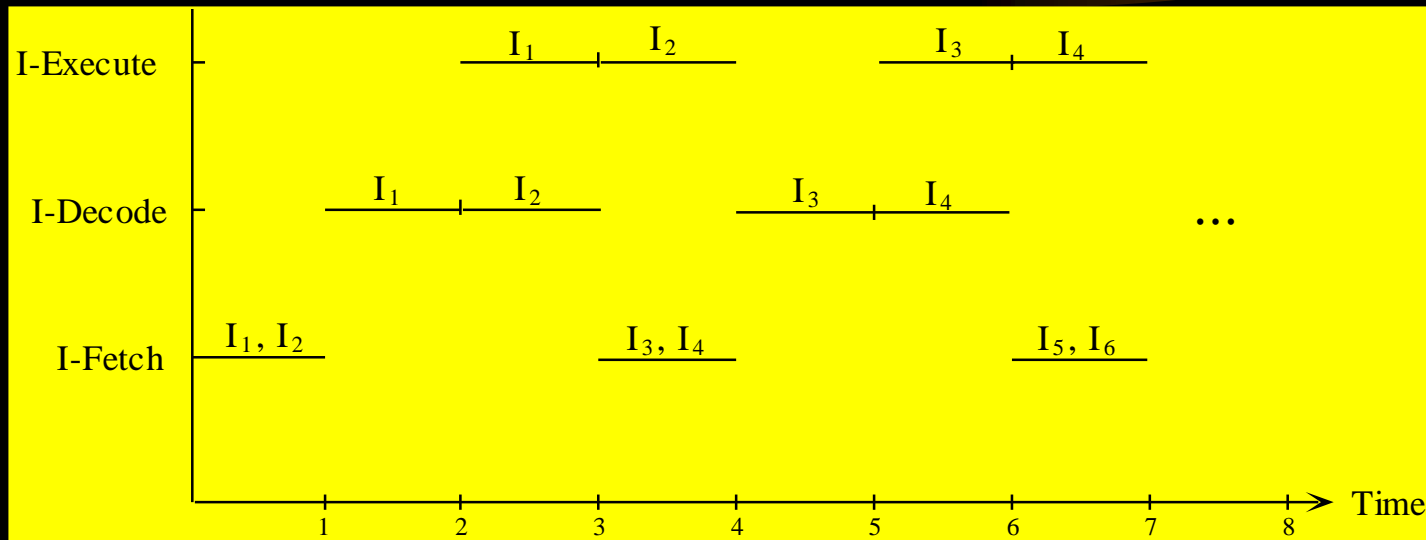


◆ Computation Gap — Logical Speed

★ Concurrency — Model-3

- This architectural model allows two instructions to be fetched at the same time - i.e., a **Primitive look-ahead** concept - while being able to overlap I-Fetch, I-Decode and I-Execute phases

◆ **Computation Gap — Logical Speed**
 ✳ **Concurrency — Model-3**



● Time to execute n instructions

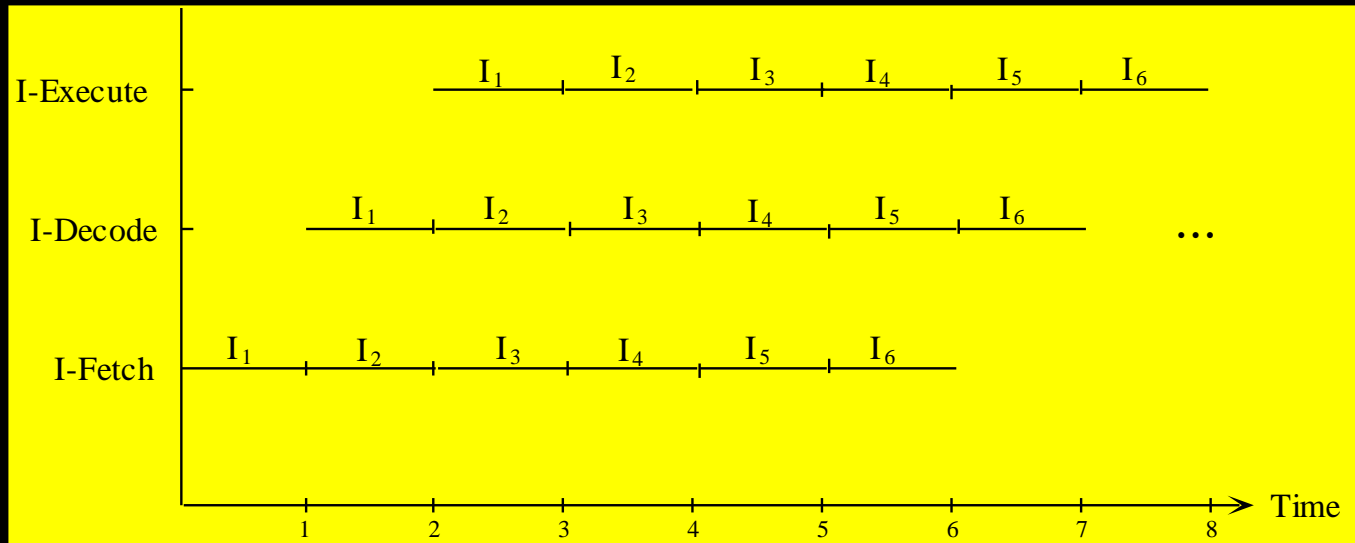
$$T_n = t + \lceil n/2 \rceil t + \lceil n/2 \rceil t + \lfloor n/2 \rfloor t \approx ((n+1) + \lfloor n/2 \rfloor) t$$

◆ Computation Gap — Logical Speed

★ Concurrency — Model-4

- In this architecture I-Fetch, I-Decode and I-Execute phases are carried out in a **pipeline** fashion

◆ Computation Gap — Logical Speed
★ Concurrency — Model-4



● Time to execute n instructions:

$$T_n = 3t + (n-1)t = (n+2)t$$

Introduction to High Performance Computer Architecture

◆ Question

- ★ Before advancing further let me pose the following question: How can we increase the performance of CDC6600?

◆ CDC 7600

★ General Philosophy

- It is an upward-compatible member of the CDC 6600 that was announced in 1969.
- It was four to eight times faster than CDC 6600.
- It had a clock period more than three times faster than clock cycle of CDC6600.
- Main memory was also more than three times faster than of CDC6600.
- The back up memory also was about two times faster than of CDC6600.

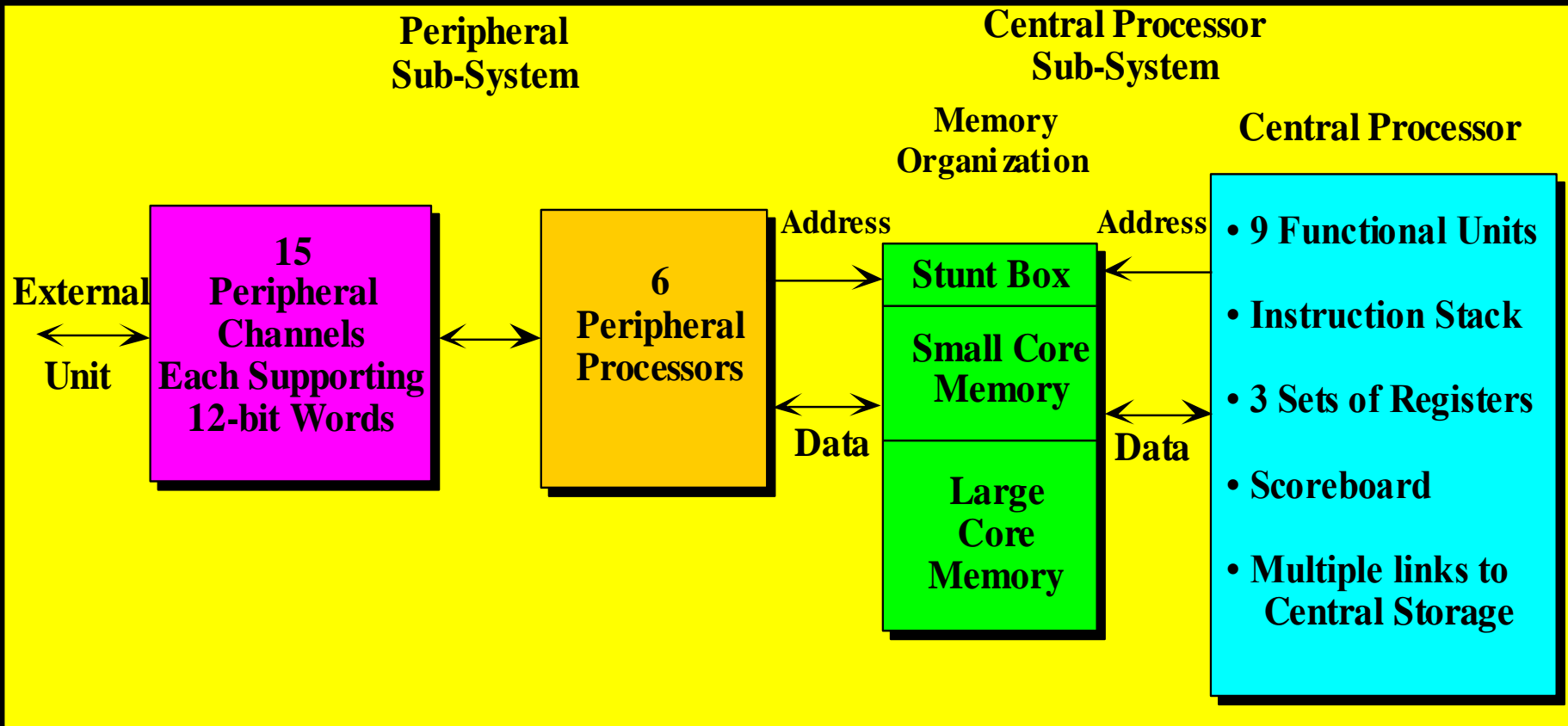
◆ CDC 7600

★ General Philosophy

- Separation of the input/output operations from the central processor operations.
- Multifunctional processor.
- Hierarchical and Interleaved memory organization.
- Pipelined instruction cycle.
- Pipelined functional units (except the divide unit).

Introduction to High Performance Computer Architecture

◆ CDC 7600



◆ CDC 7600

★ Memory Organization

- Small Core Memory (SCM) — 32 banks of 60-bit words.
- Stunt Box
- Large Core Memory (LCM) — 8 banks of 480-bit words each with 4-bit parity.

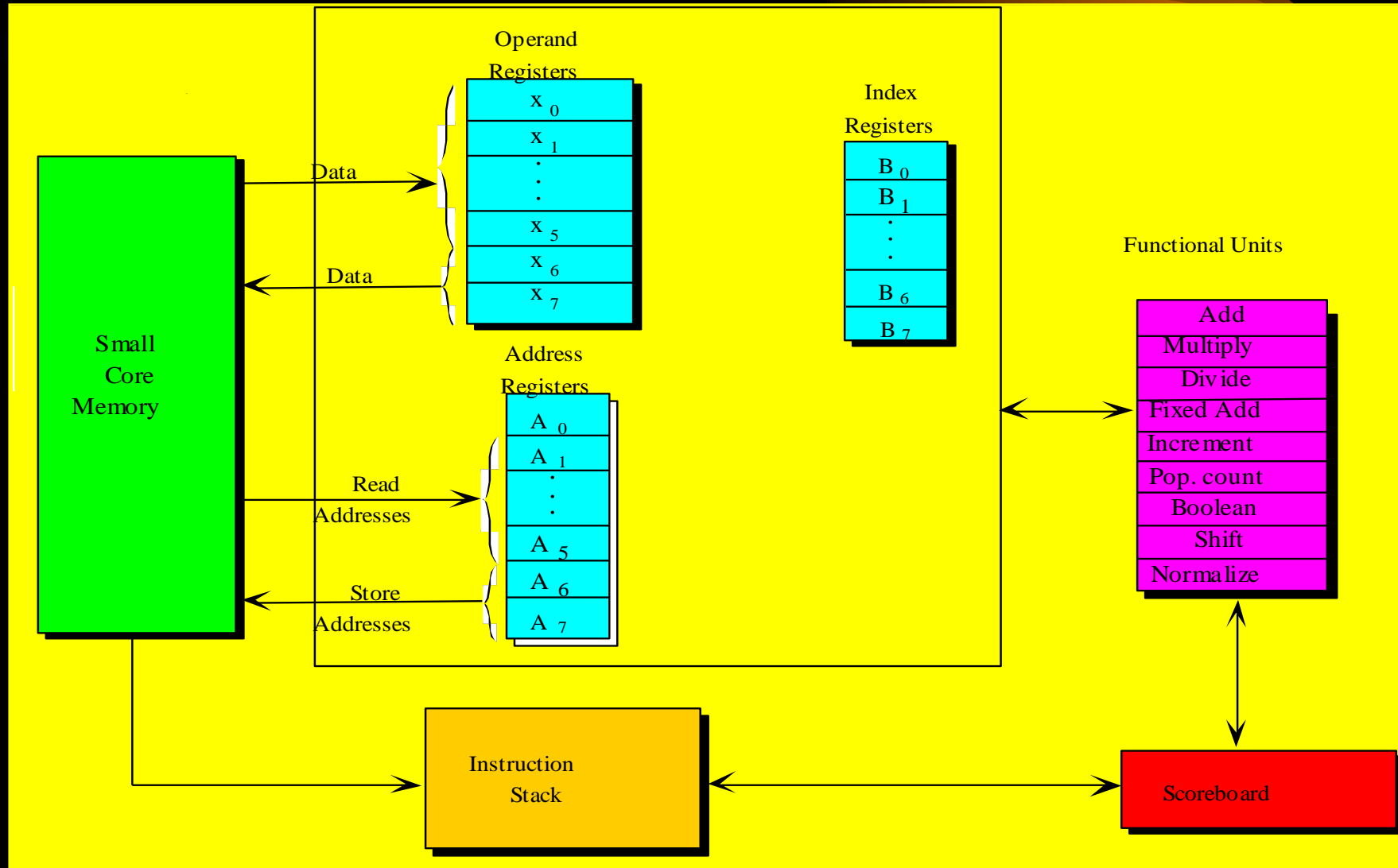
Introduction to High Performance Computer Architecture

◆ CDC 7600

★ **Central Processor** is composed of:

- 24 Registers
- Instruction Stack
- Functional Units
- Scoreboard

Introduction to High Performance Computer Architecture



◆ CDC 7600

★ 24 Registers

- Eight operand registers (X-registers) - 60 bits.
- Eight address registers (A-registers) - 18 bits.
- Eight index registers (B-registers) - 18 bits.
- Operand registers are paired up one-for-one with a corresponding address register.
- Five address-operand register pairs are used for read and two are used for write.

◆ CDC 7600

★ Instruction Stack

- Instruction Stack is a collection of twelve 60-bit registers.
- This configuration allows up to 48 previously fetched instructions to be readily available in the instruction stack.

◆ CDC 7600

★ Instruction Stack

- A reference to an instruction resident in the instruction stack will be responded to by the instruction stack. This:
 - allows faster access to the referenced instruction, and
 - reduces the main memory contention.



◆ CDC 7600

★ Functional Units

- Functional units make up the arithmetic and logic portion of the CDC 7600.
- In contrast to the CDC 6600, all the functional units are **pipelined** except the divide unit.
- Functional units are **independent** of each other and may **operate simultaneously**.

Introduction to High Performance Computer Architecture

◆ CDC 7600

★ **Functional Units:** The nine units are:

- Floating Point Add
- Floating Point Multiply
- Floating Point Divide
- Fixed Point Add
- Increment
- Boolean
- Population count
- Shift
- Normalize

Introduction to High Performance Computer Architecture

◆ CDC 7600

Unit	Function	Segment Time	Execution Time
Boolean	Basic Logic operations, Pack/unpack Flt. point	1	2
Shift	Shifting, mask generation	1	2
Fixed Add	Integer add/sub	1	2
Floating Add	Floating add	1	4
Floating Multiply	Floating Multiply	2	5
Floating Divide	Floating divide	18	20
Normalize	Normalize operations	1	3
Pop. Count	Count number of 1s in a word	1	2
Increment	1's complement add/sub	1	2

◆ CDC 7600

- Four clock cycles are required to access a SCM bank (if no conflict).
- All of the functional units, except the multiply and divide, can start a new operation in each clock cycle.



◆ Questions

- ★ Within the scope of the CDC 7600 organization, discuss the conflicts and their possible resolutions.
- ★ Fill out the entries in the following table for:
$$Y = AX^2 + BX + C$$

Introduction to High Performance Computer Architecture

Word #	Instruction	Semantics	I S S U E	S T A R T	R E S U L T	U N I T	F E T C H	S T O R E
N ₁	A ₁ = A ₁ + K ₁	FETCH X	1	1	?	?	?	
	A ₂ = A ₂ + K ₂	FETCH A	?	?	?	?	?	
N ₂	X ₀ = X ₁ * X ₁	FORM X ²	?	?	?	?		
	X ₆ = X ₀ * X ₂	FORM AX ²	?	?	?	?		
	A ₃ = A ₃ + K ₃	FETCH B	?	?	?	?	?	
N ₃	A ₄ = A ₄ + K ₄	FETCH C	?	?	?	?	?	
	X ₃ = X ₃ * X ₁	FORM BX	?	?	?	?		
	X ₅ = X ₆ + X ₃	FORM A X ² + BX	?	?	?	?		
N ₄	X ₇ = X ₅ + X ₄	FORM Y	?	?	?	?		
	A ₇ = A ₇ + K ₅	STORE Y	?	?	?	?		

◆ Computation Gap — Concurrency

★ Classification

- Different researchers made an attempt to classify the concurrent space. This includes:
 - Feng's Classification
 - Flynn's Classification
 - Handler's Classification
- In this course we will concentrate on Flynn's classification.

◆ Computation Gap — Concurrency

★ Flynn's classification

- Flynn has classified the concurrent space according to the multiplicity of instruction and data streams

I = {Single Instruction stream, Multiple Instruction stream}

D = {Single Data stream, Multiple Data stream}

◆ Computation Gap — Concurrency

★ Flynn's classification

- The cartesian product of the two aforementioned sets will define four different classes:
 - SISD
 - SIMD
 - MISD
 - MIMD



◆ Computation Gap — Concurrency

- In this course, concurrent space is classified into two groups:
 - Control Flow
 - Data Flow



◆ Computation Gap — Concurrency

- ★ In the **control flow** model of computation, execution of an instruction activates the execution of the next instruction.
- ★ In the **data flow** model of computation, availability of the data activates the execution of the next instruction(s).



◆ Computation Gap — Concurrency

- ★ Within the scope of the **control flow** systems we distinguish three sub-classes:
 - Parallel Systems
 - Pipeline Systems
 - Multiprocessors
- ★ This distinction is due to the exploitation of concurrency and the interrelationships among the control unit, processing elements and memory modules in each group.



◆ Computation Gap — Parallel Systems

- ★ These systems are the **natural extension** of parallel ALU processors, where concurrency is exploited through a collection of **identical** and **independent** processing elements controlled by the same control unit.

◆ Computation Gap — Parallel Systems

- ★ According to Flynn's classification these systems are classified as SIMD organization.
- ★ Parallel systems are **synchronous organizations, scalable**, and offer a good degree of **fault tolerance**.



◆ Computation Gap — Parallel Systems

★ Parallel systems can be further classified as:

- Ensemble Processors
- Array Processors
- Associative Processors

◆ Computation Gap — Parallel Systems

★ **Ensemble Processors** had very limited capability and flexibility. Because of these limitations this class mainly became a conceptual class.

- It is an extension of a conventional uniprocessor system.
- It is a collection of N processing elements and N memory modules under the control of a single control unit.



◆ Computation Gap — Parallel Systems

★ Ensemble Processors

- Each processing element consists of an ALU, a set of local registers and very limited local control capability.
- There exist no direct communication paths among processing elements.
- There exist fixed interconnections among processing elements and memory modules.

◆ Computation Gap — Parallel Systems

★ Array of Processors

- It is composed of N identical processing elements under the control of a single control unit and a number of memory modules.
- Processing units and memory elements communicate with each other through an interconnection network.
- Complexity of the control unit is at the same level of the uniprocessor system.



◆ Computation Gap — Parallel Systems

★ Array of Processors

- Control unit is a computer with its own high speed registers, local memory and arithmetic logic unit.
- The main memory is the aggregate of the memory modules.
- Control and scalar type instructions are executed in the control unit.
- Vector instructions are performed in the processing elements.

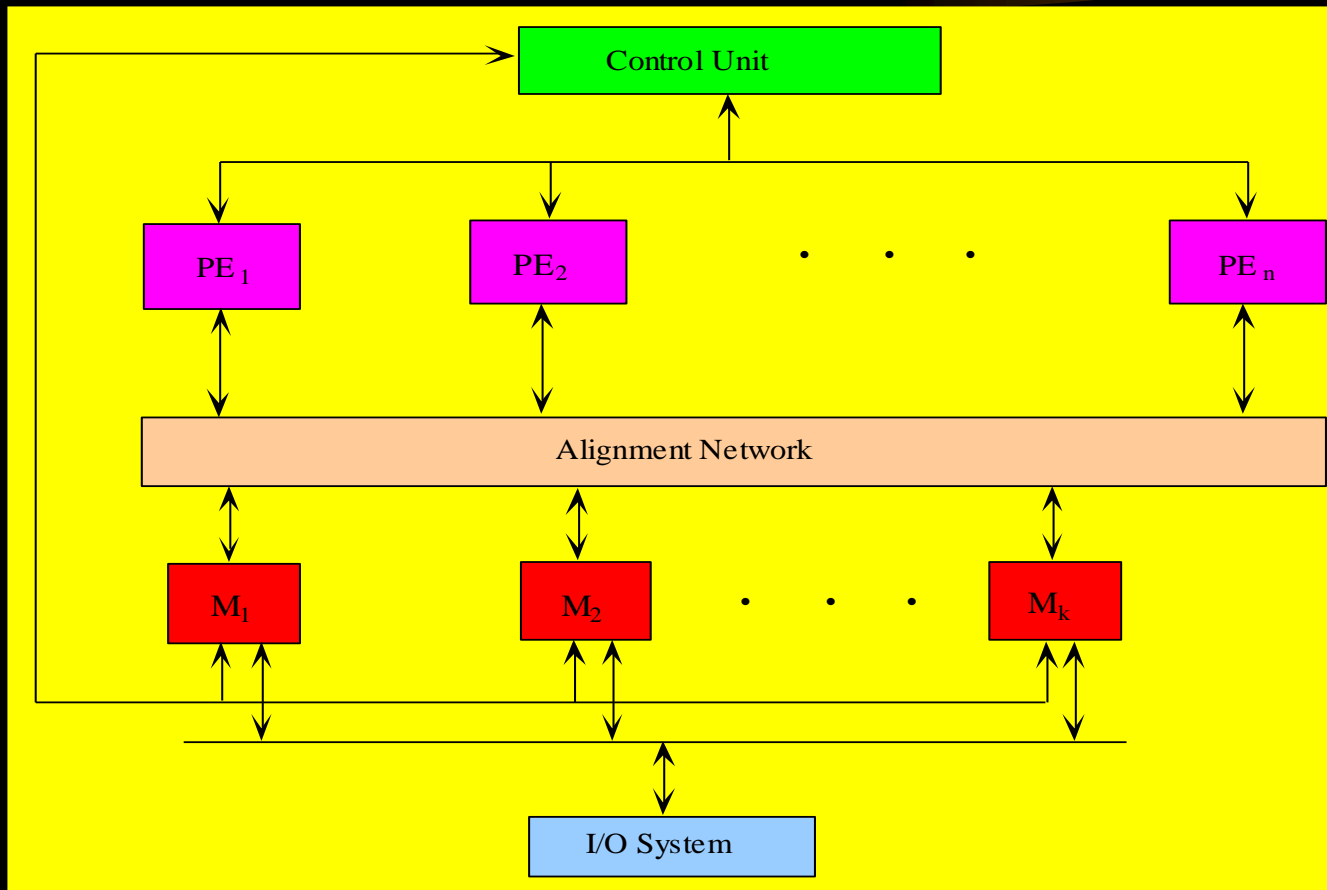
◆ Computation Gap — Parallel Systems

★ Array of Processors

- Array processors can be further classified based on complexity of the processing elements or the processors memory modules relationships:
 - Processing element complexity
 - Single-bit processors (connection machine)
 - Multi-bit processors (ILLIAC IV)
 - Processor-memory interconnection
 - Global memory organization (BSP)
 - Dedicated memory organization (ILLIAC IV)

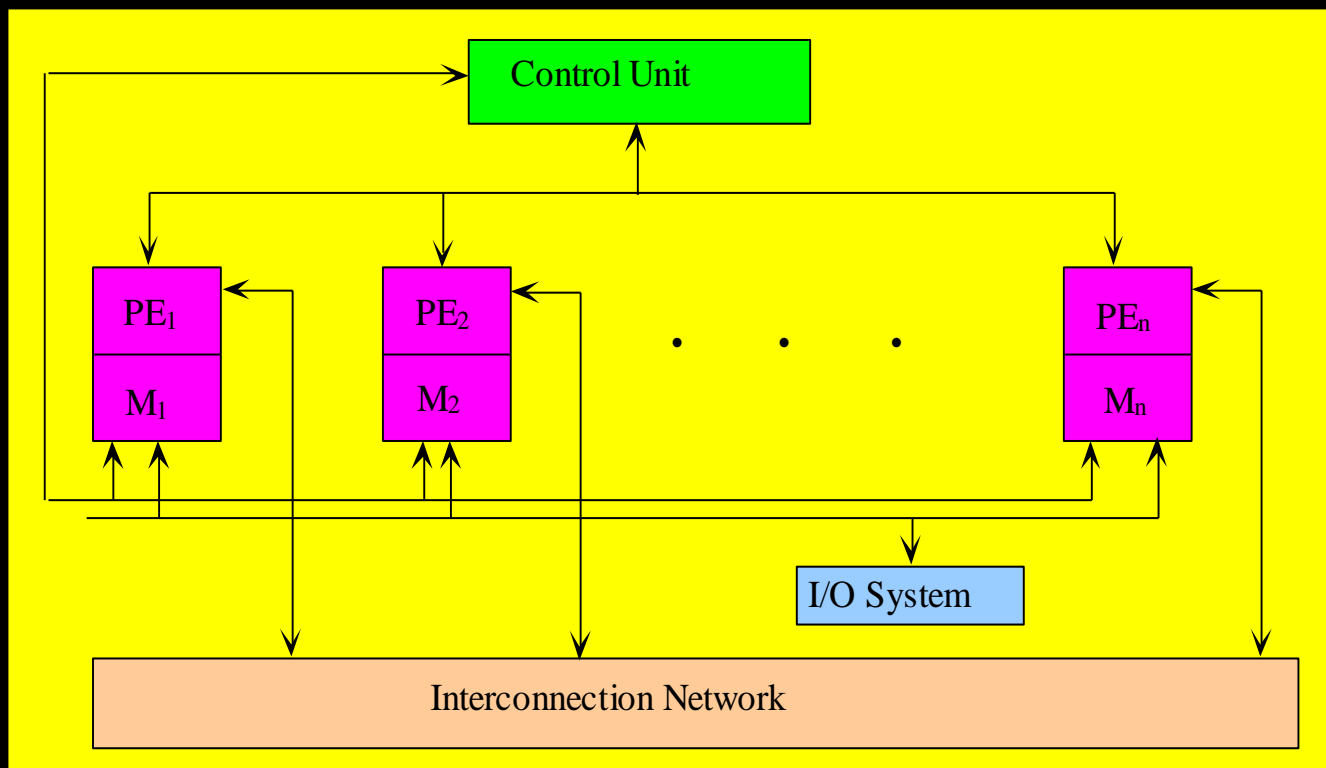
◆ Computation Gap — Parallel Systems

★ Array of Processors — Global Memory Organization



◆ Computation Gap — Parallel Systems

★ Array of Processors — Dedicated Memory Organization



◆ Questions

- ★ Compare and contrast single-bit and multi-bit array processor organizations against each other.
- ★ Compare and contrast global memory and dedicated memory array processor organizations against each other.
- ★ Discuss the problem(s) which degrade the performance of an array processor the most.

◆ Computation Gap — Parallel Systems

★ Array of Processors

- Data structuring and detection of parallelism in a program are the major bottlenecks in an array processor organization.
- Operations such as $X(i) = A(i) * B(i) \quad 1 \leq i \leq n$ could be executed in parallel, if the elements of the arrays A and B are distributed **properly** among the processors or memory modules - i^{th} processor is assigned the task of computing $X(i)$.

◆ Computation Gap — Parallel Systems

★ Array of Processors — Example

★ Compute $Y = \sum_{i=1}^N A(i) * B(i)$ where A and B are two one dimensional arrays of N elements, and elements of A and B are properly distributed among processors - assume a dedicated organization.

◆ Computation Gap — Parallel Systems

★ Array of Processors — Example

- The product terms are generated in parallel.
- Additions will be performed in $\log_2 N$ iterations.
- Speed up factor (S) then is:

$$S = \frac{2N-1}{1 + \log_2 N} \approx \left(\frac{N}{\log_2 N} \right)$$

- at the expense of a poor resource utilization(why)?



◆ Computation Gap — Parallel Systems

★ Associative Processors

- An associative processor is defined as an associative memory capable of performing arithmetic and logic operations.
- Within this scope, then, an **associative computer** is defined as a system that uses an associative memory or associative processor as an essential component for storage or processing, respectively.

◆ Computation Gap — Parallel Systems

★ Associative Processors

- The main motivations for the application of associative processing are:
 - reducing the **semantic gap** and bottleneck in the conventional systems, and
 - increasing the performance due to the parallel nature of the operations at the storage level and elimination of address computation.



◆ Computation Gap — Multiprocessor Systems

- ★ Multiprocessor systems are the nature extension of parallel systems (justify this?)
- ★ The attribute that characterizes a multiprocessor system is the **sharing** of a **global memory** by several independent processing units making up the system.

◆ Computation Gap — Multiprocessor Systems

- ★ Two arguments justify a multiprocessor organization:
 - **Higher throughput:** due to the ability to overlap both computation intensive and I/O intensive tasks among independent processors in the system.
 - Existence of a large class of problems that can be split up into a number of smaller and independent tasks.

◆ Computation Gap — Multiprocessor Systems

- ★ A multiprocessor system contains two or more processing units, each with its own control unit. Processors can be **homogeneous** or **heterogeneous**.
- ★ Processing units are **not highly specialized**.
- ★ Processing units **share** a main memory that usually consists of several independently accessible modules.



◆ Computation Gap — Multiprocessor Systems

- ★ Besides a common memory, processing units usually share other resources such as I/O channels and devices.
- ★ The whole system is under the control of a **single integrated operating system.**

◆ Computation Gap — Multiprocessor Systems

★ Multiprocessor systems can be grouped into two classes:

- **Tightly Coupled:** shared memory modules are separated from processors by an interconnection network or a multiport interface.
- The memory access time (assuming no conflict) is independent of the module being accessed (**uniform Memory access**).



◆ Computation Gap — Multiprocessor Systems

- Loosely Coupled: each processor has a local-public memory.
- Each processor can directly access its memory module, but all other accesses to non-local memory modules must be made through an interconnection network (non-uniform memory access).



◆ Computation Gap — Multiprocessor Systems

- ★ Besides the higher throughput, multiprocessor systems offer **more reliability** since failure in any one of the redundant components can be tolerated through system **reconfiguration**.



◆ Computation Gap — Multiprocessor Systems

- ★ Multiprocessor organization is a logical extension of the parallel system - i.e., array of processor organization.
- ★ However, the degree of freedom associated with the processors are much higher than it is in an array processor.



◆ Computation Gap — Multiprocessor Systems

- ★ The independence of the processors and the sharing of resources among the processors - both desirable features - are achieved at the expense of an **increase in complexity** at both the hardware and software levels.



◆ Computation Gap — Pipeline Systems

- ★ The term pipelining refers to a design technique that introduces concurrency by taking a basic function to be involved repeatedly in a process and partitioning it into several sub-functions with the following properties:

◆ Computation Gap — Pipeline Systems

- Evaluation of the basic function is equivalent to some sequential evaluation of the sub-functions.
- Other than the exchange of inputs and outputs, there is no interrelationships between sub-functions.
- Hardware may be developed to execute each sub-function.
- The execution time of these hardware units are usually approximately equal.



◆ Computation Gap — Pipeline Systems

★ The concept of pipelining can be implemented at different levels. With regard to this issue, one can then address:

- Arithmetic Pipelining
- Instruction Pipelining
- Processor Pipelining



◆ Computation Gap — Pipeline Systems

★ Pipeline systems can be further classified as:

- Linear Pipe
- Feedback Pipe
- Scalar Pipe
- Vector Pipe
- Unifunction Pipe
- Multifunction Pipe
- Statically Configured Pipe
- Dynamically Configured Pipe

◆ Computation Gap — Pipeline Systems

★ Example

- Calculate the speed-up factor for in a multifunction pipe of 5 stages, where A and B are two one dimensional arrays of N elements.

$$Y = \sum_{i=1}^N A(i) * B(i)$$

two one

◆ Computation Gap — Pipeline Systems

★ Example

- Product terms will be generated in $(n-1)+5$ steps.
- Additions will be performed in
 $5+(\lceil n/2 \rceil - 1) + 5+(\lceil n/4 \rceil - 1) + \dots + 5+(1-1) \cong (4\log_2 n + n)$ steps.
- Speed-up ratio

$$S = \frac{5(2n-1)}{2n+4 \log_2 n+4} \approx 5 \quad \text{for large } n$$

◆ Parallelism vs. Pipelining

- Both techniques attempt to increase the performance.
- Parallelism is achieved through the **replication** of basic hardware, while pipelining is the result of **staging** the hardware unit.
- In general, parallelism is more **reliable** than pipelining.
- Parallelism is more **extendable** than pipelining.
- The difference between the two schemes also shows up in **memory organization, bandwidth, internal interconnection and control.**

◆ Shortcomings of the Concurrent Control Flow Systems

- **Complexity:** This is mainly due to the simultaneous competition/cooperation of several modules over common resources. This leads to more complexity and sophistication at the **control structure** and **interconnection network**.
- **Specialization:** Control flow concurrent systems require specialized and **different programming skills** for efficient resource utilization.

◆ Shortcomings of the Concurrent Control Flow Systems

- **Semantic Gap:** Control flow concurrent systems offer a wider semantic gap than their sequential predecessors.
- Lack of suitable **parallel algorithms** for various applications.
- Lack of suitable **parallel high level languages** to allow the programmer to express parallelism explicitly in the problem being encoded.



◆ Shortcomings of the Concurrent Control Flow Systems

- Lack of suitable **compilation techniques** to detect embedded parallelism in a sequential program.
- Lack of suitable **control algorithms** to distribute hardware resources among concurrently running programs.

◆ Shortcomings of the Concurrent Control Flow Systems — Example

★ Let us compute

$$y = \sum_{i=1}^N A(i)*B(i)$$

◆ Shortcomings of the Concurrent Control Flow Systems — Example

- In **Parallel System** assuming that the operands are properly aligned and only a subset of processor which handle these operands become active at successive iterations:
 - Product terms could be generated in parallel in one step.
 - Additions will be performed in $\log_2 N$ iterations.
 - Speed up ratio $S = \frac{2N-1}{1 + \log_2 N} \approx 0 \left(\frac{N}{\log_2 N} \right)$ is achieved at the expense of poor **resource utilization**.

◆ Shortcomings of the Concurrent Control Flow Systems — Example

- In **Pipeline System** assuming a multi-function pipe of 5 stages and availability of a constant flow of data to the pipe:

- Product terms will be generated in $(N-1)+5$ steps.
- Additions will be performed in

$$5 + \left(\left\lceil \frac{N}{2} \right\rceil - 1\right) + 5 + \left(\left\lceil \frac{N}{4} \right\rceil - 1\right) + \dots + 5 + (1-1) \leq 4\log N + N \text{ steps}$$

- Speed up ratio

$$S = \frac{5(2N-1)}{2N+4\log N+4} \approx 5$$

◆ Sample Problems

- Explain the following terms:
 - Carry lookahead
 - Iterative method (multiplication)
 - Interleaved memory

◆ Sample Problems

- The "add and shift" algorithm can be used to multiply two negative numbers (say A and B) in 2^s complement format.
 - Calculate the correction term.
 - Apply your conclusion to perform the following operation using "add and shift" algorithm (show step-by-step operation).

$$\begin{array}{r} 101001 \\ * 110011 \\ \hline \end{array}$$

Note: numbers are in 2^s complement format.

◆ Sample Problems

- Use **SRT** division method to perform the following operation: AQ/B
 - where $AQ = .00100000$ and $B = .0110$
- Show step-by-step operation.

◆ Sample Problems

- Apply the Column Compression technique to perform the following operation:

$$101011 * 110101$$

Note: numbers are in 2^s complement format.

- Explain what factors degrade the performance of an interleaved memory. Why

◆ Sample Problems

- For a fully parallel word organized associative memory, write an algorithm to find the minimum word in the memory array. Note: each word contains one unsigned number. Your algorithm should be well documented, and you have to show me that your algorithm is correct.

◆ Sample Problems

- Within the scope of cache organization:
 - explain write-back and write-through policies,
 - discuss the advantages and disadvantages of each, and
 - is it possible to extend the write-back policy for better performance? How? Explain.
- Discuss the advantages of having an instruction cache (which only stores the program code but not data).

◆ Sample Problems

- Apply reduction of summands scheme (using half and full adders) to perform the following operation:
 - $1101111 * 0110011$
 - Calculate the execution time of the operation (show the work).
Note: operands are in 2^s complement format.
- Explain the following terms:
 - pipelining
 - parallelism
 - non-restoring division
 - size gap

◆ Sample Problems

- Name and explain at least 5 distinct factors which affect the length and the format of an instruction (assembly (machine) level). Why?
- The following operation is assumed in which numbers are in 2^s complement format:
 - $1100111 * 0101111$
 - Calculate the correction factor and apply Hurson's scheme to perform the aforementioned multiplication.

◆ Sample Problems

- In an interleaved memory organization prove that in case of branch the average number of memory modules to be used effectively is:

$$IB_n = \frac{1 - (1 - \lambda)^n}{\lambda}$$

Where n is the number of modules and λ is the probability of a successful branch.

◆ Sample Problems

- Within the scope of cache organization explain the following terms:
 - Direct Mapping,
 - Associative Mapping, and
 - Set Associative Mapping.
- compare and contrast the aforementioned mapping schemes against each other.

◆ Sample Problems

- Explain the following terms (be as specific and precise as possible):
 - Computation Gap,
 - Multiprocessor System,
 - RISC (Reduced Instruction Set Computer), and
 - Parallel System.

◆ Sample Problems

- Consider CDC 6600 (multi-function organization): Table 1 shows the timing table in such an organization for an expression. Fill out the entries and explain your reasoning.

Introduction to High Performance Computer Architecture

Word #	Instruction	Semantics	ISSUE	START	RESULT	UNIT	FETCH	STORE
N ₁	A ₁ =A ₁ +k ₁	Fetch X (long)	1	1	4	5	9	
	A ₂ =A ₂ +k ₂	Fetch A (long)	?	3	6	7	?	
N ₂	A ₃ =A ₃ +k ₃	Fetch B (long)	9	9	12	13	17	
	A ₄ =A ₄ +k ₄	Fetch C (long)	11	11	14	15	19	
N ₃	B ₁ =B ₀ +1	Set B ₁ to 1 (long)	?	17	20	21		
	B ₂ =B ₀ +k ₅	Set vector length (long)	19	19	22	23		
N ₄	X ₀ =X ₁ *X ₁	Form X ² (short)	25	25	35	36		
	X ₆ =X ₀ *X ₂	Form AX ² (short)	26	35	?	46		
	X ₀ =X ₃ *X ₁	Form BX (short)	?	36	46	47		
	A ₁ =A ₁ +B ₁	Fetch Next X (short)	37	37	40	41	?	
N ₅	B ₂ =B ₂ -B ₁	Decrement B ₂ (short)	41	41	44	45		
	X ₅ =X ₆ +X ₀	Form AX ² +BX (short)	42	?	50	51		
	X ₇ =X ₅ +X ₄	Form Y (short)	?	51	55	56		
	A ₇ =A ₇ +B ₁	Store Y (short)	?	?	59	60		64