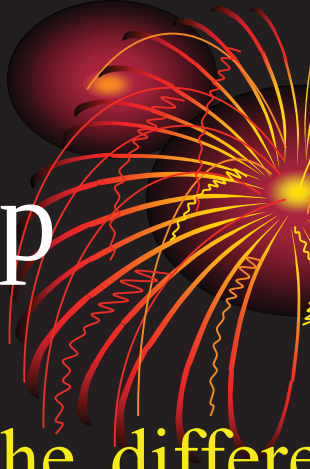
Computation Gap
Instruction Level Parallelism

A.R. Hurson
Computer Science Department
Missouri Science & Technology
hurson@mst.edu

Computation Gap



- Computation gap is defined as the differe between computational power demanded the application environments computational capability of the exis computers.
- Today, one can find many applications wh require orders of magnitude m computations than the capability of the called super-computers and super-systems.

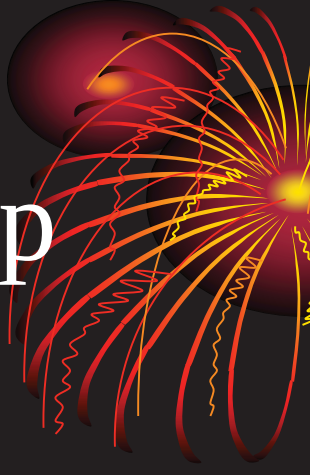
Computation Gap

- Some Applications
  - It is estimated that the so called Prob
    Solving and Inference Systems require
    environment with the computational powe
    the order of 100 MLIPS to 1 GLIPS (1 LIP
    100-1000 instructions).

Computation Gap

- Some Applications
  - Experiences in Fluid Dynamics have sh
    that the conventional super-computers
    calculate steady 2-dimensional flow in minu
  - However, conventional super-compu
    require up to 20 hours to handle time depen
    2-dimensional flow or steady 3-dimensi
    flows on simple objects.

Computation Gap



- Some Applications
  - Numerical Aerodynamics Simulator require environment with a sustained speed of 1 bi FLOPS.
  - Strategic Defense Initiative requires distributed, fault tolerant compu environment with a processing rate of MOPS.

Computation Gap

- Some Applications
  - U.S. Patent Office and Trademark has a databas size 25 terabytes subject to search and update.
  - An angiogram department of a mid-size hos generates more than $64 * 10^{11}$ bits of data a year.
  - NASA's Earth Observing System will generate than 11,000 terabytes of data during the 15-year t period of the project.

Computation Gap

- Performance

CDC	STAR-100	25-100	MFLOI
DAP		100	MFLOI
ILLIAC IV		160	MFLOI
HEP		160	MFLOI
CRAY-1		25-80	MFLOI
CRAY X-MP(1)		210	MFLOI
CRAY X-MP(4)		840	MFLOI
CRAY-2		250	MFLOI
CDC	CYBER 200	400	MFLOI
Hitachi	S-810(10)	315	MFLOI
Hitachi	S-810(20)	630	MFLOI
Fujitsu	FACOM VP-50	140	MFLOI
Fujitsu	FACOM VP-100	285	MFLOI
Fujistu	FACOM VP-200	570	MFLOI
Fujistu	FACOM VP-400	1,140	MFLOI

Computation Gap

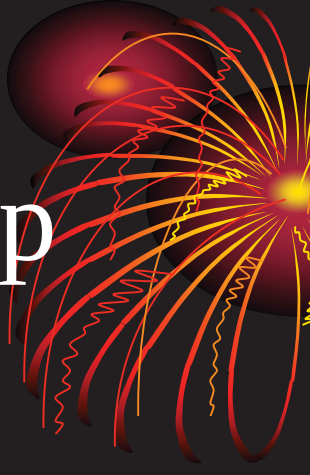
- Performance

NEC	SX-1	570	MFLOPS
NEC	SX-2	1,300	MFLOPS
IBM	RP3	1,000	MFLOPS
MPP	8-bit integer	1,545-6,553	MIPS
MPP	12-bit integer	795-4,428	MIPS
MPP	16-bit integer	484-3,343	MIPS
MPP	32-bit FL	165-470	MIPS
MPP	40-bit FL	126-383	MIPS

Computation Gap

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

Computation Gap

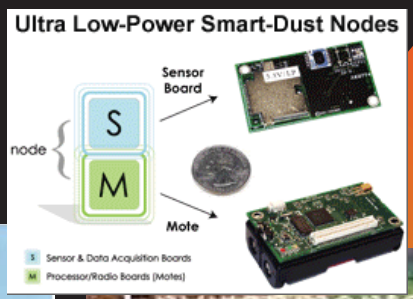


- Some New Applications
  - A recent estimate puts the amount of new inform generated in 2002 to be 5 exabytes (1 exabyte= 10^{18} which is approximately equal to all words spoken by h beings) and 92% of this information is in hard disk. W good fraction of this information is of transient int useful information of archival value will continu accumulate.
  - The TREC database holds around 800 million static having 6 trillion bytes of plain text equal to the size million books.
  - The Google system routinely accumulates millions of of new text information every week.

Computation Gap

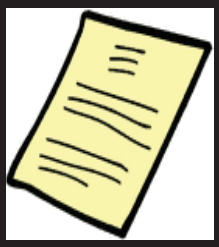
Sensors

Multimedia

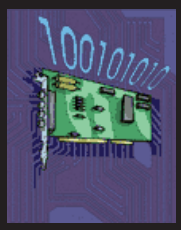


Image

Text



Binary



1970 1980 1990 2000

Computation Gap

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

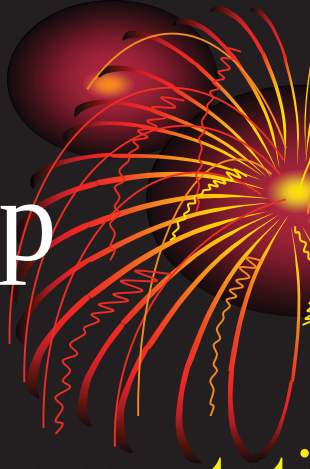
$$\frac{25 * 10^{12}}{10^{6}} = 25 * 10^{6} \text{ sec.} \approx 4 * 10^{5} \text{ min.} \approx 7 * 10^{3} \text{ Hours} \approx 290 \text{ da}$$

  - Even if the performance is improved by a factor of 1000, it takes about 8 hours to exhaustively search this database!

Computation Gap

- Problem
  - NOT PRACTICAL!
  - WHAT ARE THE SOLUTIONS?

Computation Gap

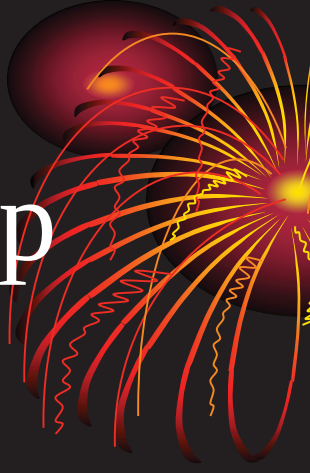


- Reduce the amount of needed computati (advances in software technology algorithms).
- Improve the speed of the computers:
  - Physical Speed (Advances in hardv technology).
  - Logical Speed (Advances in comp architecture/organization).

Computation Gap

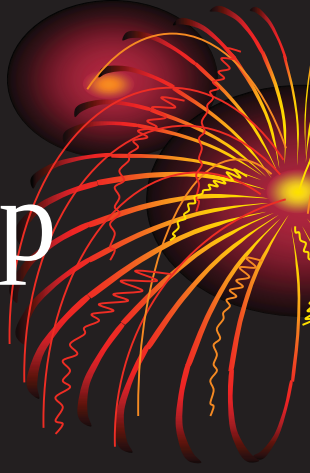
- Architectural Advances of the U processor Organization
  - Organization of the conventional uni-proce systems can be modified in order to remove existing bottlenecks. For example, Access is one of the problems in the von Neum organization.

Computation Gap



- Access Gap
  - Access gap is defined as the time differe between the CPU cycle time and the m memory cycle time.
  - Access gap problem was created by advances in technology.

Computation Gap

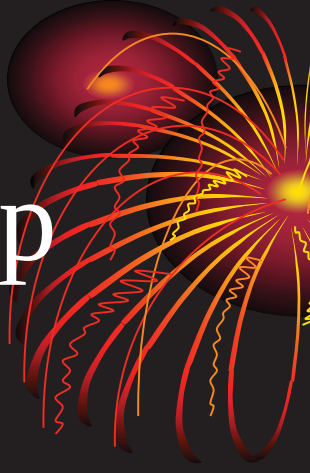


- Access Gap
  - In early computers such IBM 704, CPU and memory cycle time were identical — i.e., 12 μsec.
  - IBM 360/195 had the logic delay of 5 ηsec per s the CPU cycle time of 54 ηsec and the main mer cycle time of .756 μsec and CDC 7600 had the and main memory cycle time of 27.5 ηsec and μsec, respectively.

Computation Gap

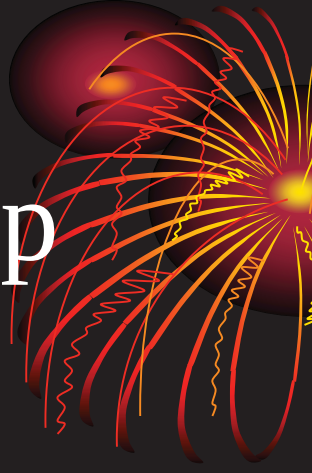
- System Architecture/Organization
  - To overcome the technological limitati computer designers have long been attracte techniques that are classified under the ter "Concurrency".

Computation Gap



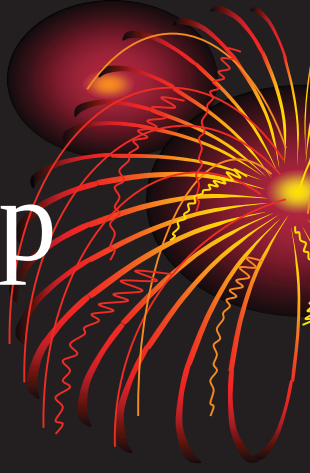
- Concurrency
  - Concurrency is a generic term which def the ability of the computer hardware simultaneously execute many actions at instant.
  - Within this general term are several recognized techniques such as Parallel Pipelining, and Multiprocessing.

Computation Gap



- Concurrency
  - Although these techniques have the same or
    and are often hard to distinguish, in prac
    they are different in their general approach.

Computation Gap



- Concurrency
  - Parallelism achieves concurrency replicating/duplicating the hardware struc many times,
  - Pipelining takes the approach of splitting function to be performed into smaller pie allocating separate hardware to each piece, overlapping operations of each piece.

Computation Gap

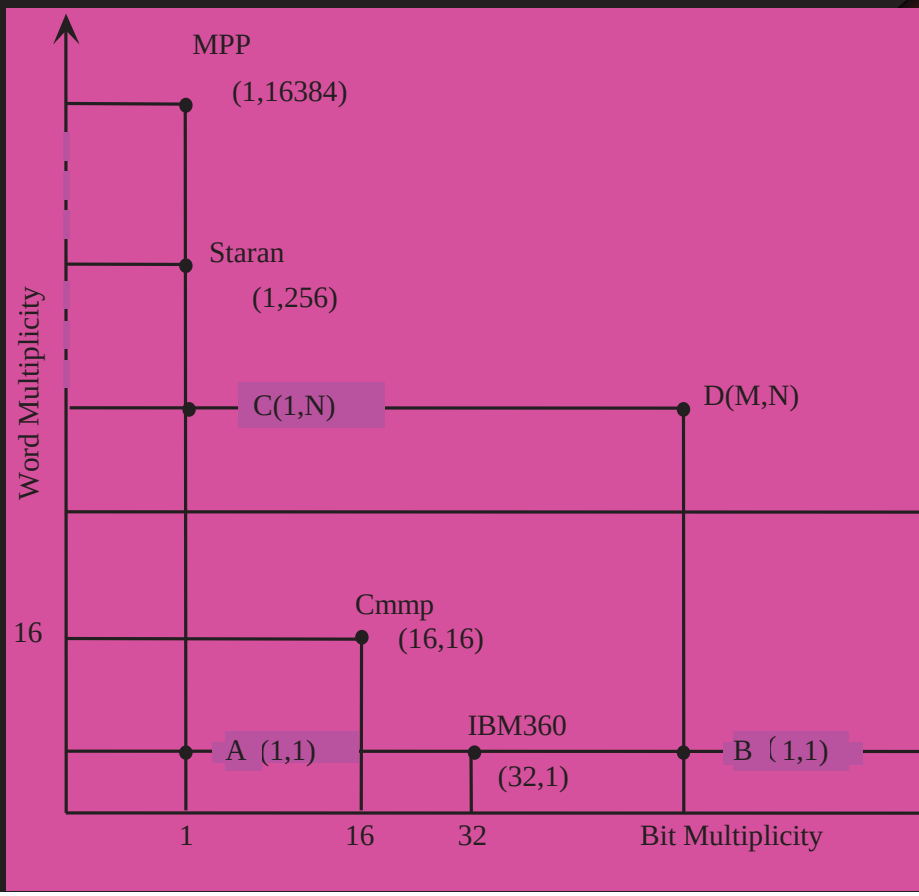
- Concurrent Systems
  - Classification
    - Feng's Classification
    - Flynn's Classification
    - Handler's Classification

Computation Gap

- Concurrent Systems
  - Feng's Classification
    - In this classification, the concurrent spac identified as a two dimensional space based o bit and word multiplicities.

Computation Gap

- Concurrent Systems
  - Feng's Classification



Computation Gap

- Concurrent Systems
  - Feng's Classification
    - Point *A* represents a pure sequential machine - i uni-processor with serial *ALU*.
    - Point *B* represents a uni-processor with pa *ALU*.
    - Point *C* represents a parallel bit slice organizatio
    - Point *D* represents a parallel word orga organization.

Computation Gap

- Concurrent Systems
  - Flynn's Classification
    - Flynn has classified the concurrent space acco to the multiplicity of instruction and data stream

I= { Single Instruction Stream (SI), Multiple Instruction Stream (M

D={Single Data Stream (SD), Multiple Data Stream (MD) }

Computation Gap

- Concurrent Systems
  - Flynn's Classification
    - The Cartesian product of these two sets will d four different classes:
      - SISD
      - SIMD
      - MISD
      - MIMD

Computation Gap

- Concurrent Systems
  - Flynn's Classification — Revisited
    - The MIMD class can be further divided based o
      - Memory structure — global or distributed
      - Communication/synchronism mechanism — s variable or message passing.

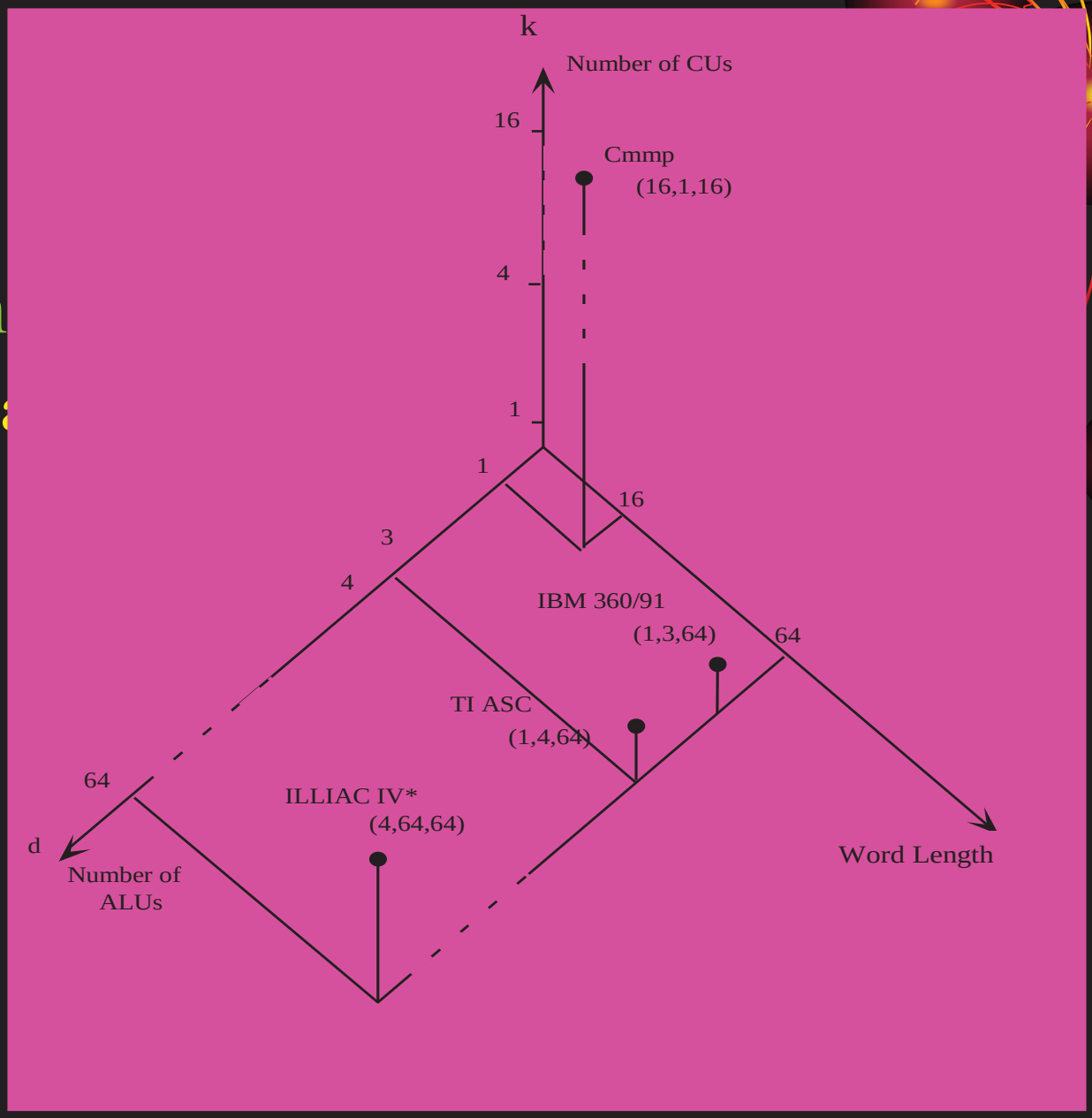
Computation Gap

- Concurrent Systems
  - Flynn's Classification — Revisited
    - As a result we have four additional classe computers:
      - GMSV — Shared memory multiprocessors
      - GMMP — ?
      - DMSV — Distributed shared memory
      - DMMP — Distributed memory (multi-computers)

Computation Gap

- Concurrent Systems
  - Handler's Classification
    - Handler has extended Feng's concurrent space third dimension, namely, the number of co units.
    - Handler's space is defined as T=(k,d,w):
      - *k* number of control units,
      - *d* number of *ALU*s controlled by a control unit,
      - *w* number of bits handled by an *ALU*.

- Con
  - H


k
Number of CUs
16
Cmmp
(16,1,16)
4
1
1
16
3
4
IBM 360/91
(1,3,64)
64
TI ASC
(1,4,64)
64
ILLIAC IV*
(4,64,64)
d
Number of
ALUs
Word Length

Computation Gap

- Concurrent Systems
  - Handler's Classification
    - Point (1,1,1) represents von Neumann machine serial ALU.
    - Point (1,1,M) represents von Neumann mac with parallel ALU.

Computation Gap

- Concurrent Systems
  - Handler's Classification
    - To represent pipelining at different levels - macro pipeline, instruction pipeline and arithm pipeline - diversity, sequentiality, flexibility/adaptability, the original Hand scheme has been extended by three varia (k′,d′,w′) and three operators (+, *, v).

Computation Gap

- Concurrent Systems
  - Handler's Classification
    - k´ represents macro pipeline
    - d´ represents instruction pipeline
    - w´ represents arithmetic pipeline
    - + represents diversity (parallelism)
    - * represents sequentiality (pipelining)
    - v represents flexibility/adaptability

Computation Gap

- Concurrent Systems
  - Handler's Classification
    - According to the extended Handler's scheme:

CDC_{7600}: (1*1, 1*9, 60*1)

CDC_{Star}: (1*1, 2*1, 64*4)

ILLIAC IV: (1*1, 1*1, 48*1) * (1*1, 64*1, 64*1)

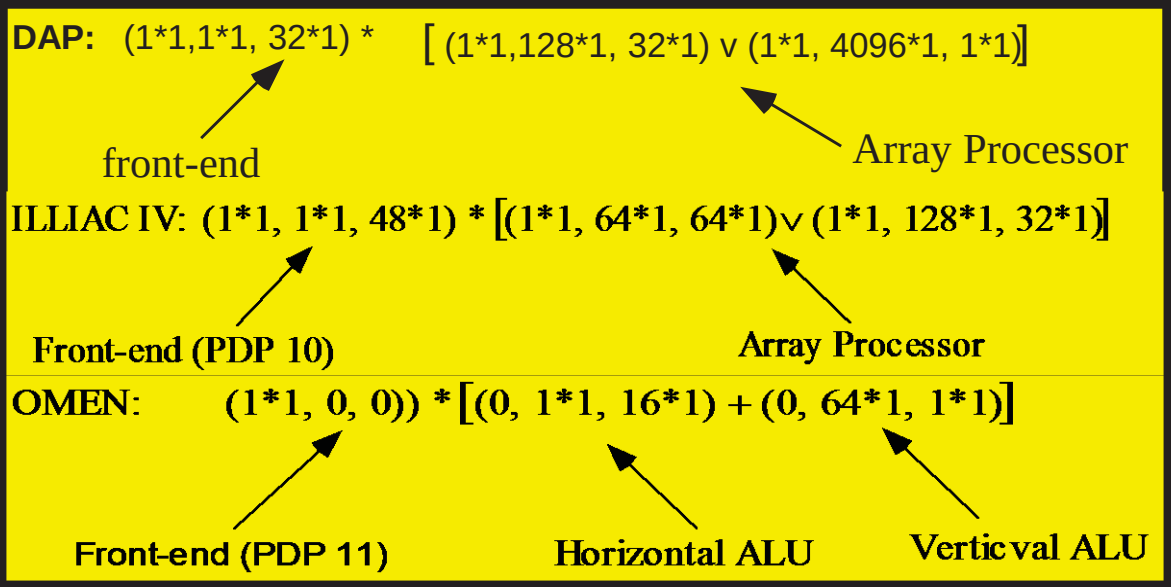
Front-end (B 6700) — Array Processor

CDC_{7600}: (15*1, 1*1, 12*1) * (1*1, 1*9, 60*1)

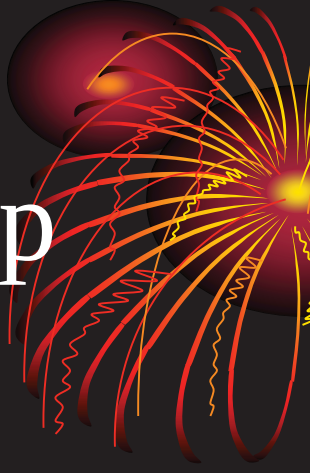
Peripheral Processors — Central Processor

Computation Gap

- Concurrent Systems
  - Handler's Classification
    - According to the extended Handler's scheme:

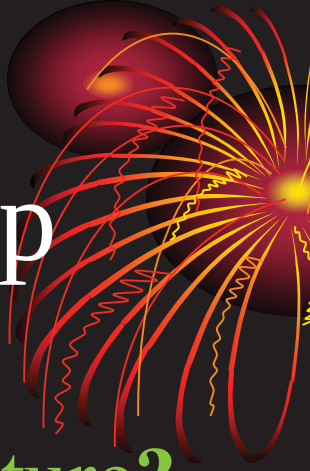


Computation Gap



- Questions
  - What are the motivations behind classification of the computer systems?
  - What are the shortcomings of aforementioned classification schemes?
  - Can you propose a new classification schem

Computation Gap

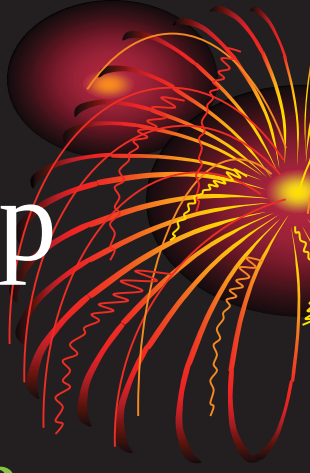


- Why classify computer architecture?
  - Generalize and identify the characteristic different systems.
  - Group machines with common architect features:
    - To study systems easier.
    - To transfer solutions easier.

Computation Gap

- Why classify computer architecture?
  - Better estimate the weak and strong points system:
    - To utilize a system more effectively.
  - Anticipate the future trends and the developments:
    - Research directions.

Computation Gap

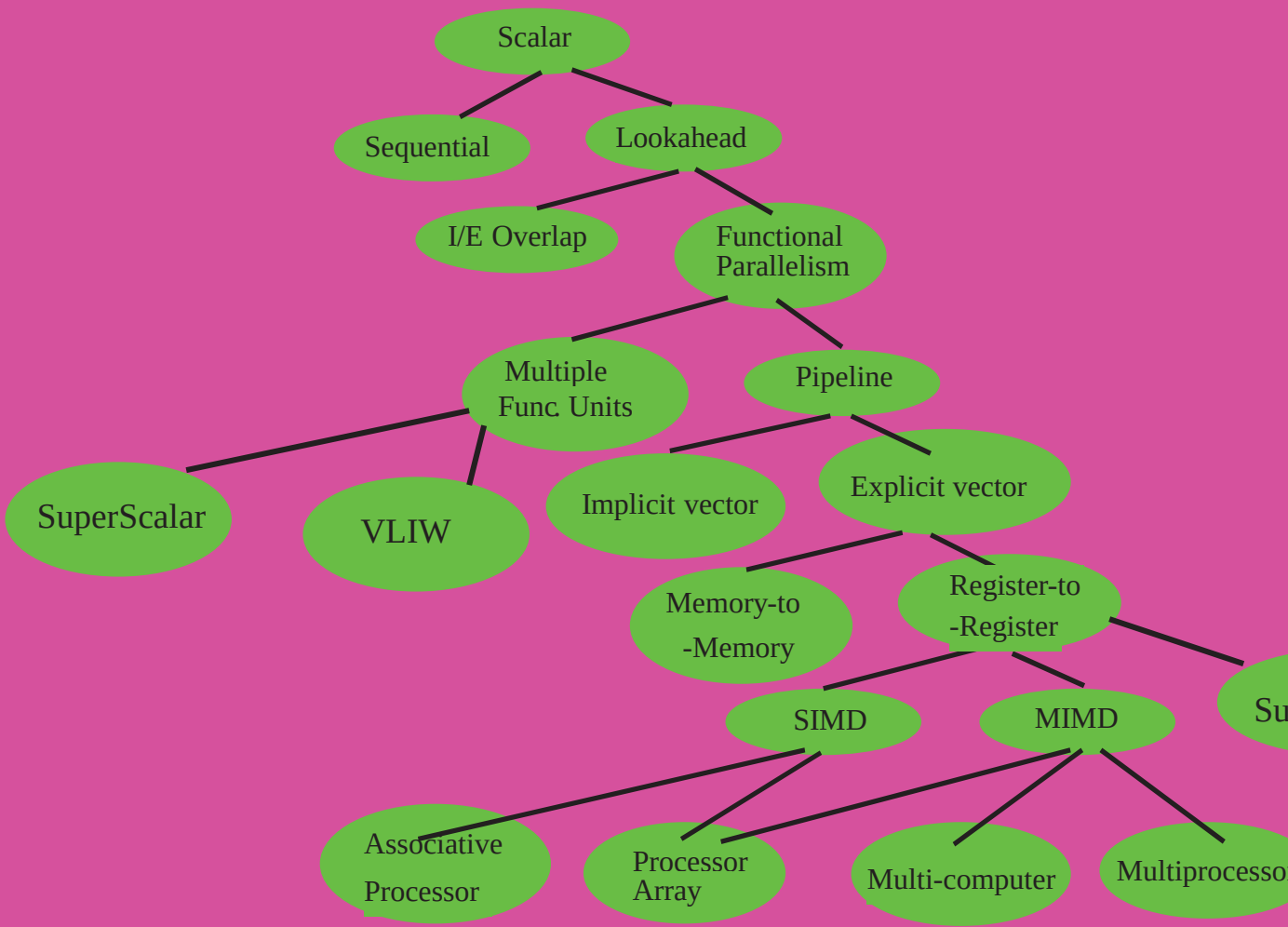


- Goals of a classification scheme
  - Categorize all existing and foreseeable desig
  - Differentiate different designs.
  - Assign an architecture to a unique class.

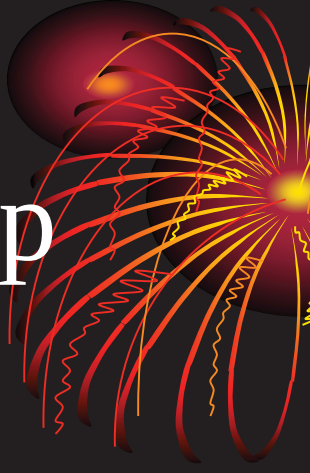
Computation Gap

- Summary
  - Computation Gap
  - How to reduce Computation Gap:
    - Advances in Software and Algorithms
    - Advances in Technology
    - Advances in Computer Organization/Architecture
  - Concurrency
  - Classification
    - Feng
    - Flynn/Extended MIMD
    - Handler

Computation Cap

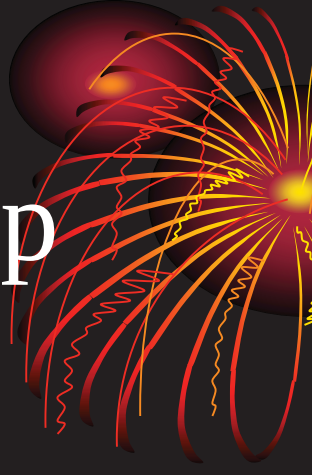


Computation Gap



- Concurrent Systems
  - We group concurrent systems into two grou
    - Control Flow
    - Data Flow

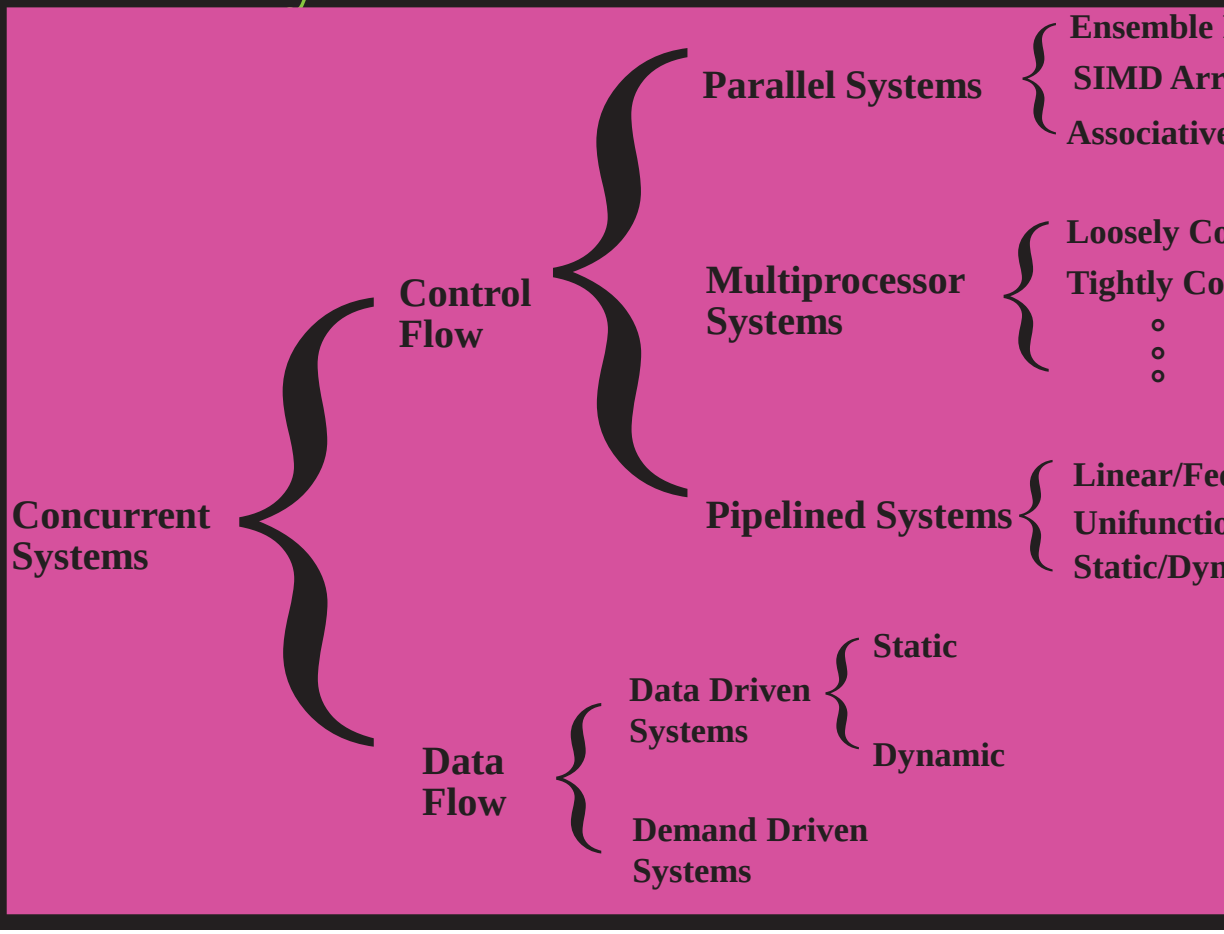
Computation Gap



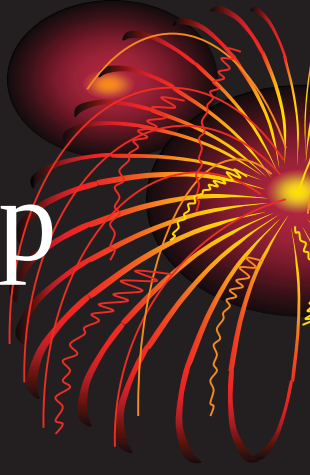
- Concurrent Systems
  - In the control flow model of computat execution of an instruction activates execution of the next instruction.
  - In the data flow model of computat availability of the data activates the execu of the next instruction(s).

Computation Gap

- Concurrent Systems



Computation Gap



- Concurrent Systems
  - Within the scope of the control flow syst
    we distinguish three classes — namely:
    - Parallel Systems
    - Pipeline Systems
    - Multiprocessors

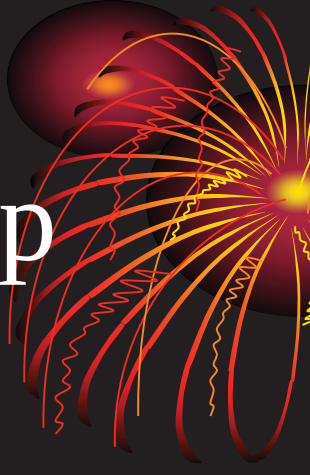
Computation Gap

- Concurrent Systems
  - This distinction is due to the exploitation concurrency and the interrelationships am the control unit, processing elements memory modules in each group.

Computation Gap

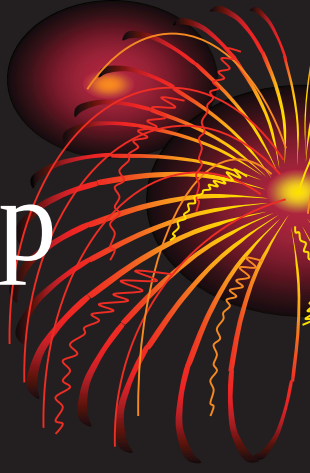
- Multiprocessor Systems
  - Multiprocessor systems can be grouped two classes:
    - Tightly Coupled (Central Memory — not scalab
    - Loosely Coupled (Distributed Memory — scala

Computation Gap



- Multiprocessor Systems
  - Tightly Coupled (Central Memory — not Scala
    shared memory modules are separated from proce
    by an interconnection network or a multiport interfa
  - All processors have equal access time to all mer
    words. Therefore, the memory access time (assu
    no conflict) is independent of the module b
    accessed (C.mmp, HEP, Encore's Multimax, Cedar
    NYU Ultracomputer).

Computation Gap

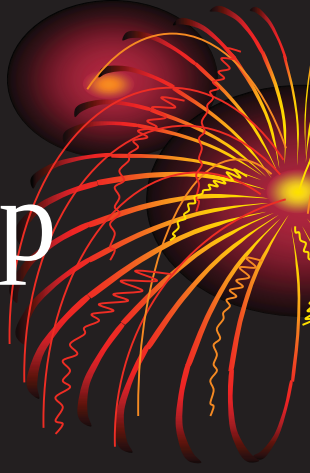


- Multiprocessor Systems
  - Loosely Coupled (Distributed Memory — Scala each processor has a local-public memory.
  - Each processor can directly access its memory mo but all other accesses to non-local memory mo must be made through an interconnection network access time varies with the location of the me module (Cm*, BBN Butterfly, and Dash).

Computation Gap

- Multiprocessor Systems
  - Besides the higher throughput, multiproce systems offer more reliability since failure in one of the redundant components can be toler through system reconfiguration.
  - Multiprocessor organization is a logical exten of the parallel system — i.e., array of proce organization. However, the degree of free associated with the processors are much hi than it is in an array of processor.

Computation Gap

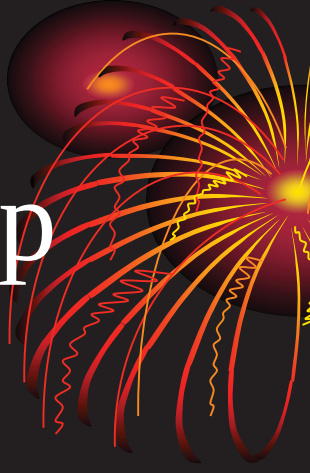


- Multiprocessor Systems
  - The independence of the processors and sharing of resources among the processors both desirable features — are achieved at expense of an increase in complexity at both hardware and software levels.

Computation Gap

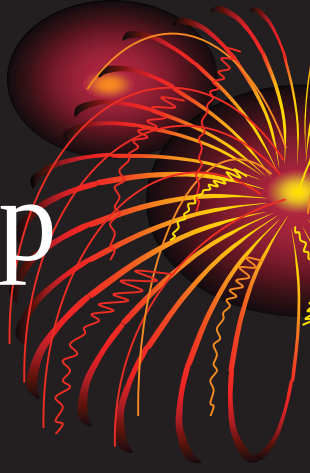
- Multi-computer Systems
  - A multi-computer system is a collection processors, interconnected by a mess passing network.
  - Each processor is an autonomous comp having its own local memory communicating with each other through message passing network (iPSC, nCUBE, Mosaic).

Computation Gap



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

Computation Gap

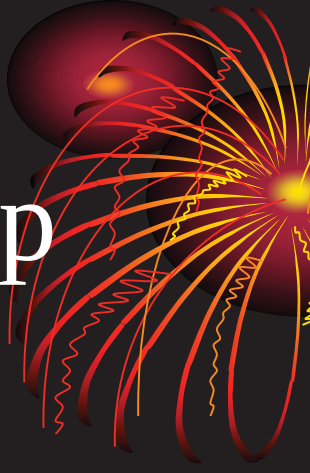


- Pipeline Systems
  - Evaluation of the basic function is equivale
    some sequential evaluation of the sub-functions.
  - Other than the exchange of inputs and outputs,
    is no interrelationships between sub-functions.
  - Hardware may be developed to execute each
    function.
  - The execution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

Computation Gap

- Pipeline Systems
  - Under the aforementioned conditions, the sp
    up from pipelining equals the number of
    stages.
  - However, stages are rarely balanced
    furthermore, pipelining does involve s
    overhead.

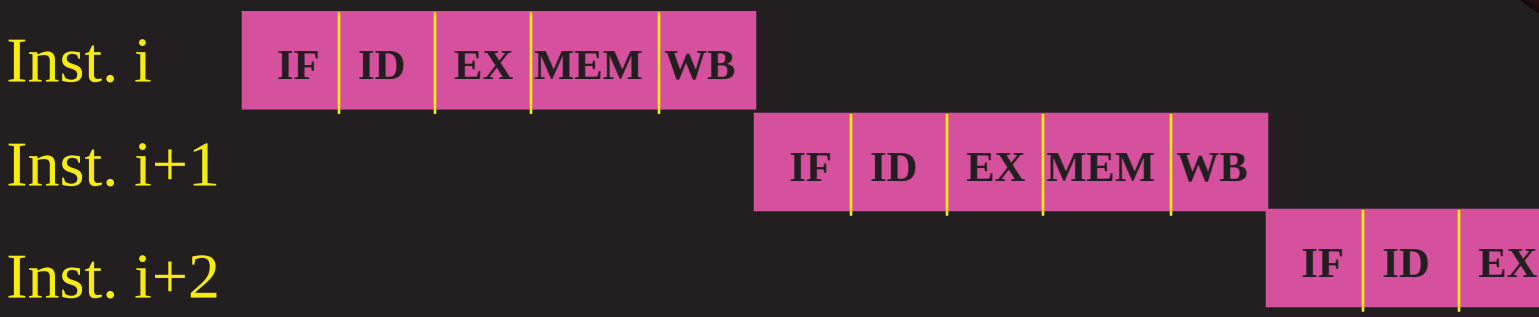
Computation Gap



- Pipeline Systems
  - The concept of pipelining can be impleme at different levels. With regard to this is one can address:
    - Arithmetic Pipelining
    - Instruction Pipelining
    - Processor Pipelining

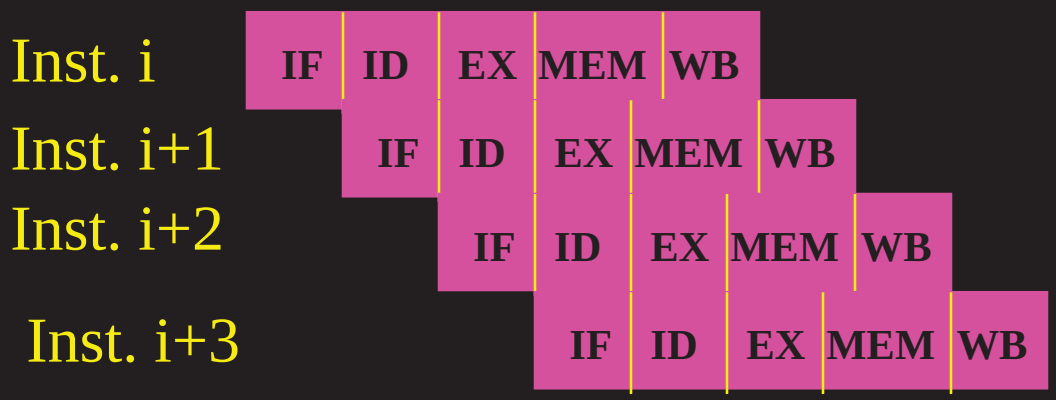
Computation Gap

- Pipeline Systems
  - Non-pipelined instruction cycle:



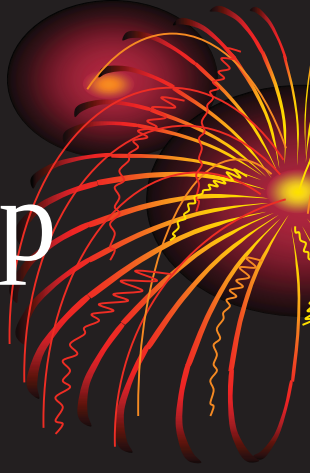
Computation Gap

- Pipeline Systems
  - Pipelined instruction cycle:



A pipelined instruction cycle gives a peak performa
of one instruction every step.

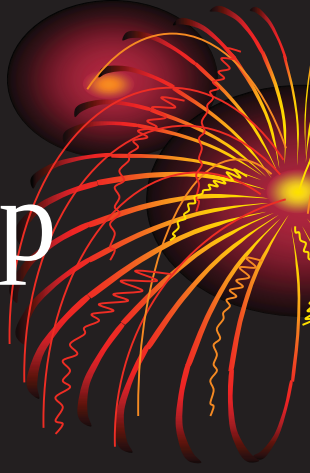
Computation Gap



- Pipeline Systems — Example
  - Assume an unpipeline machine has a 10 ns c cycles. It requires four clock cycles for the ALU branch operations and five clock cycles for the mer reference operations. Calculate the average instru execution time, if the relative frequencies of t operations are 40%, 20%, and 40%, respectively.

Ave. instr. exec. time = 10 * [(40%+20%) * 4 + 40%
= 44 ns

Computation Gap

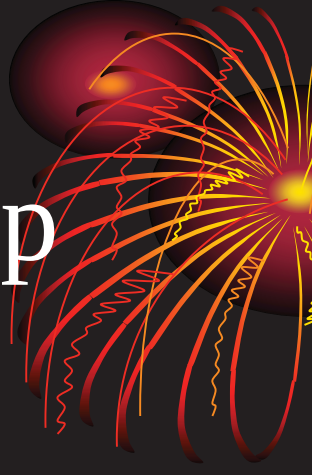


- Pipeline Systems — Example
  - Now assume we have a pipeline version of machine. Furthermore, due to the clock skew an up, pipelining adds 1 ns overhead to the clock Ignoring the latency, now calculate the ave instruction execution time.

    Ave. instr. exec. time = 10 + 1 ns, and

    Speed up = 44/11 = 4

Computation Gap



- Pipeline Systems — Example
  - Assume that the time required for the five units i instruction cycle are, 10 ns, 8 ns, 10 ns, 10 ns, and Further, assume that pipelining adds 1 ns overh Find the speed up factor:

    Ave. instr. exec. $time_{unpipeline}$ = 10 + 8 + 10 + 10 + 7
    = 45 ns

    Ave. instr. exec. $time_{pipeline}$ = 11 ns

    Speed up = 45/11 = 4.1

Computation Gap

- Pipeline Systems
  - Issues of concern:
    - Overlapping operations should not over co resources — Every pipe stage is active on e clock cycle.
    - All operations in a pipe stage should comple one clock and any combination of operations sh be able to occur at once.

Computation Gap

- Pipeline Systems
  - Pipeline systems can be further classified as
    - Linear Pipe / Feedback Pipe
    - Scalar Pipe / Vector Pipe
    - Uni-function Pipe / Multifunction Pipe
    - Statically/Dynamically Con-figured Pipe

Computation Gap

- Pipeline Systems
  - Uni-function Pipeline: Pipeline is dedicated specific function — CRAY-1 has 12 dedic pipelines.
  - Multifunction Pipeline: Pipeline system perform different functions either at different t or at the same time — TI-ACS has multifunction pipelines reconfigurable for a va of arithmetic operations.

Computation Gap

- Pipeline Systems
  - Static Pipeline: Pipeline system assumes configuration at a time.
  - Dynamic Pipeline: Pipeline system all several functional configurations to e simultaneously.

Computation Gap

- Pipeline Systems — Example
  - In a multifunction pipe of 5 stages calculate speed-up factor for

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

Computation Gap

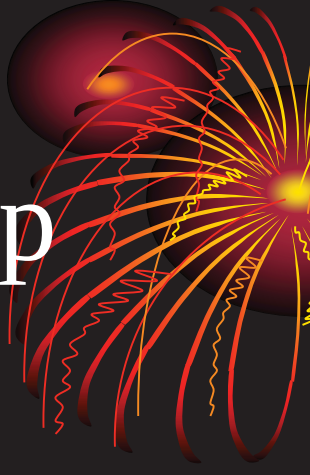
- Pipeline Systems — Example
  - Product terms will be generated in (n-1) steps.
  - Additions will be performed in:

    $5 + (\lceil n/2 \rceil - 1) + 5 + (\lceil n/4 \rceil - 1) + \ldots + 5 + (1-$

    $(4\log_2 n + n)$ steps.
  - Speed-up ratio

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

Computation Gap

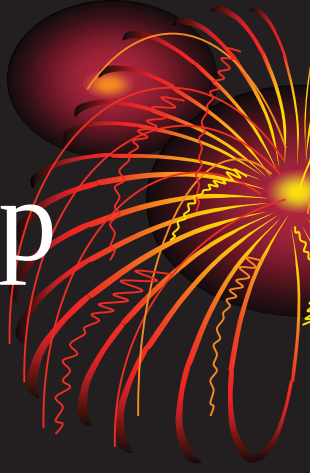


- Pipeline Systems
  - A concept known as hazard is a major con in a pipeline organization.
  - A hazard prevents the pipeline from accep data at the maximum rate that the staging c might support.

Computation Gap

- Pipeline Systems
  - A hazard can be of three types:
    - Structural Hazard: Arises from resource conflicts
      the hardware cannot support all possible combinatio
      instructions in simultaneous overlapped execution —
      different pieces of data attempt to use the same sta
      the same time.
    - Data-Dependent Hazard: Arises when an instru
      depends on the result of a previous instruction —
      pass through a stage is a function of the data value.

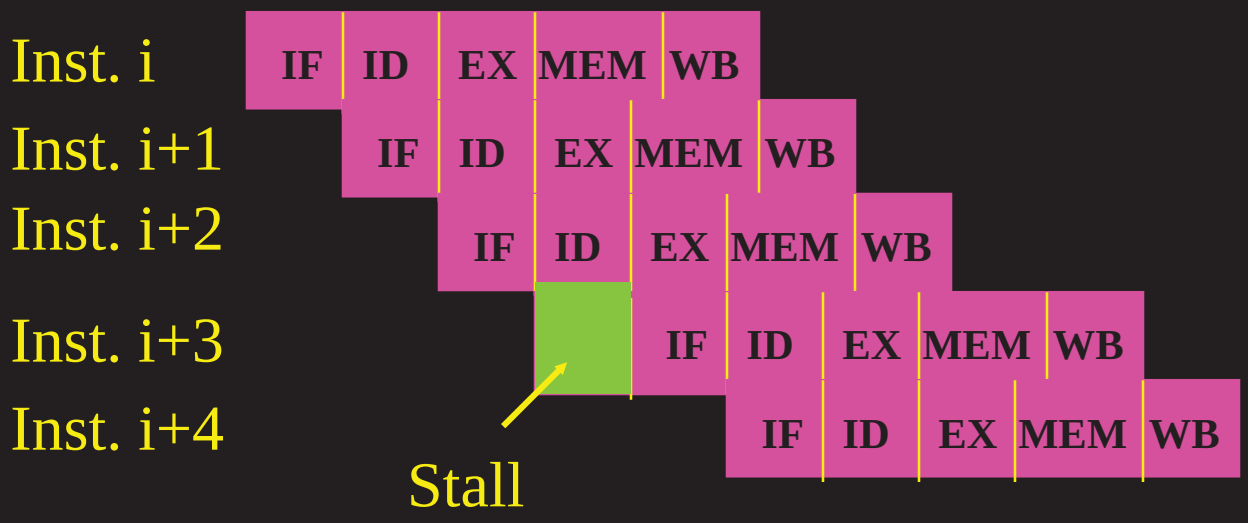
Computation Gap



- Pipeline Systems
  - Control Hazard: Arises from the pipelinin instructions that affect PC — Branch.

Computation Gap

- Pipeline Systems
  - Structural Hazard (Assume a Single mem pipeline system)



Computation Gap

- Pipeline Systems
  - Data Hazard
    - A data hazard is created whenever there dependence between instructions, and they are enough that the overlap caused by pipelining w change the order of access to an operand.

ADD	R_1, R_2, R_3
SUB	R_4, R_1, R_5

Computation Gap

- Pipeline Systems
  - Data Hazard
    - Data Hazard can be resolved with a si forwarding technique — If the forwarding hard detects that the previous ALU operation has wr to a source register of the current ALU opera control logic selects the forwarded result as the input rather the value read from the register file.

Computation Gap

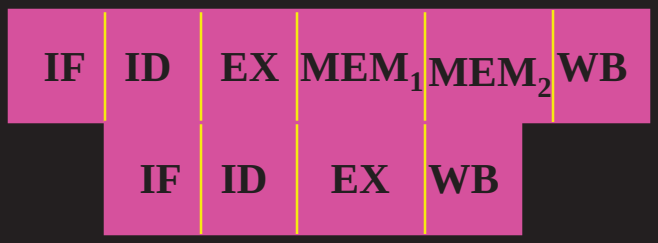
- Pipeline Systems
  - Data Hazard — Classification
    - Assume *i* and *j* are two instructions and *j* is successor of *i*, then one could expect three typ data hazard:
      - Read after write (RAW)
      - Write after write (WAW)
      - Write after read (WAR)

Computation Gap

- Pipeline Systems
  - Data Hazard — Classification
    - Read after write (RAW) — *j* reads a source befor writes it (flow dependence).
    - Write after write (WAW) — *j* writes into the sa destination as *i* does (output dependence).

LW R_1, 0(R_2)

Add R_1, R_2, R_3

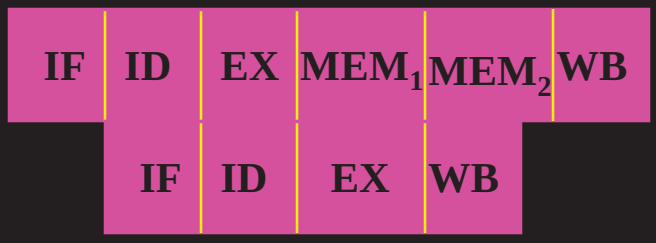


Computation Gap

- Pipeline Systems
  - Data Hazard — Classification
    - Write after read (WAR) — *j* writes into a source (anti dependence).

SW 0(R_1), R_2

Add R_2, R_4, R_3



Computation Gap

- Pipeline Systems
  - Data Hazard — Forwarding
    - One can use the concept of data forwardin overcome stall (s) due to data hazard.

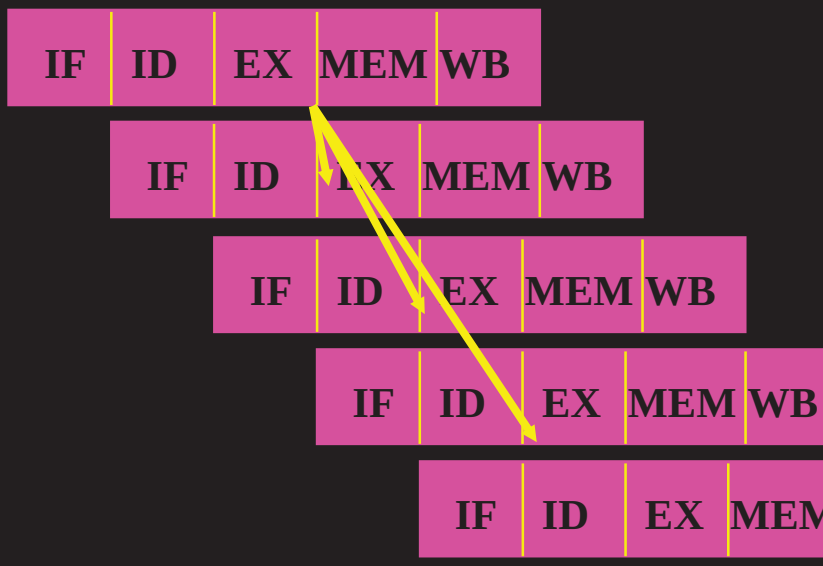
Add R_1, R_2, R_3

Sub R_4, R_1, R_5

And R_6, R_1, R_7

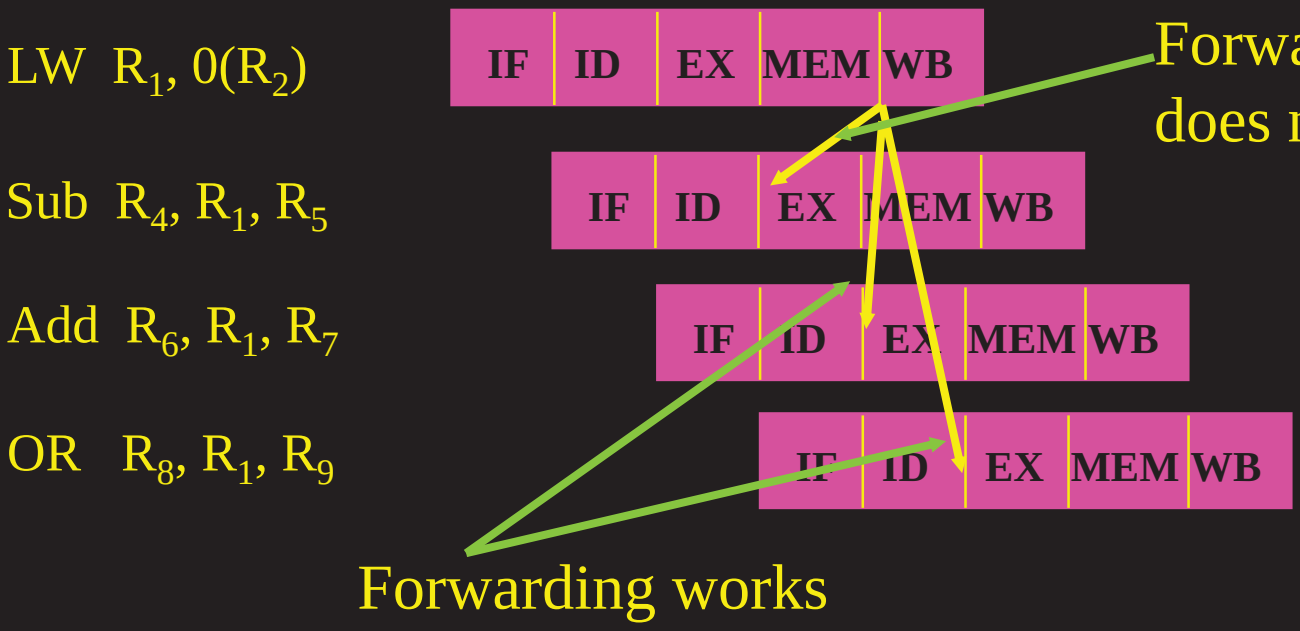
OR R_8, R_1, R_9

XOR R_{10}, R_1, R_{11}



Computation Gap

- Pipeline Systems
  - Data Hazard — Forwarding
    - In some cases data forwarding does not work:



Computation Gap

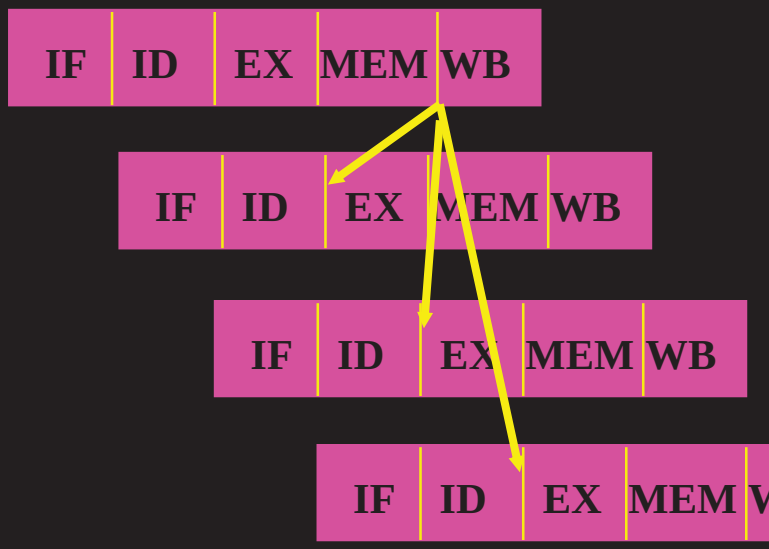
- Pipeline Systems
  - Data Hazard — Stalling
    - In cases where data forwarding does not work pipe has to be stalled:

LW R_1, A

Add R_4, R_1, R_7

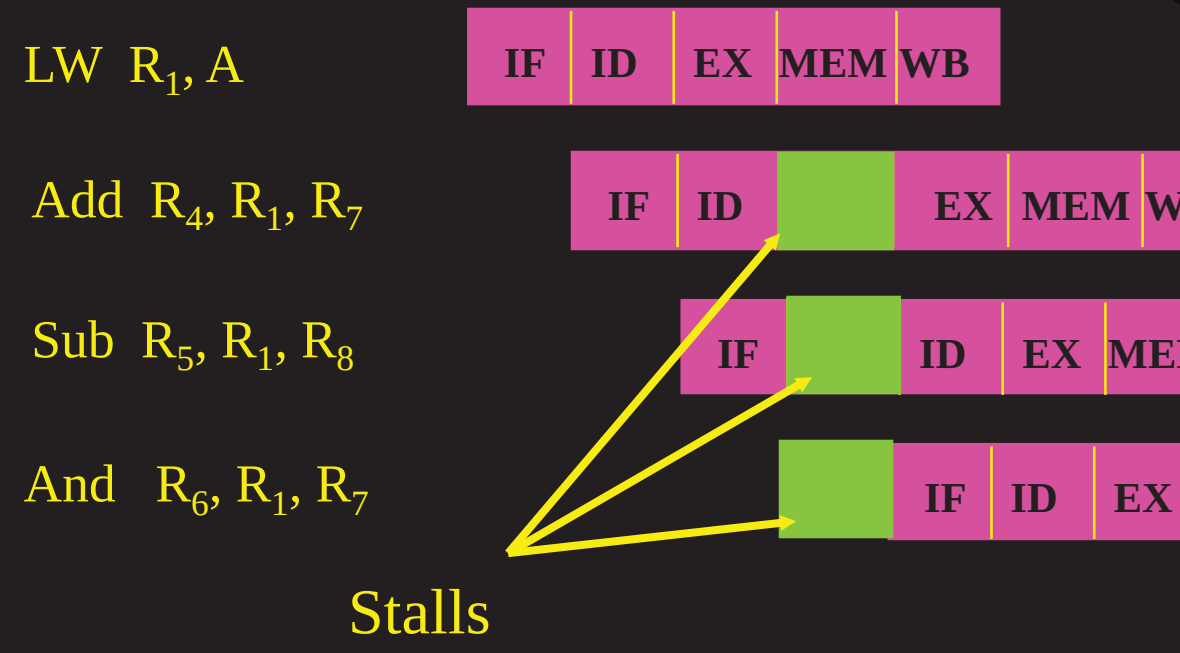
Sub R_5, R_1, R_8

And R_6, R_1, R_7



Computation Gap

- Pipeline Systems
  - Data Hazard — Stalling

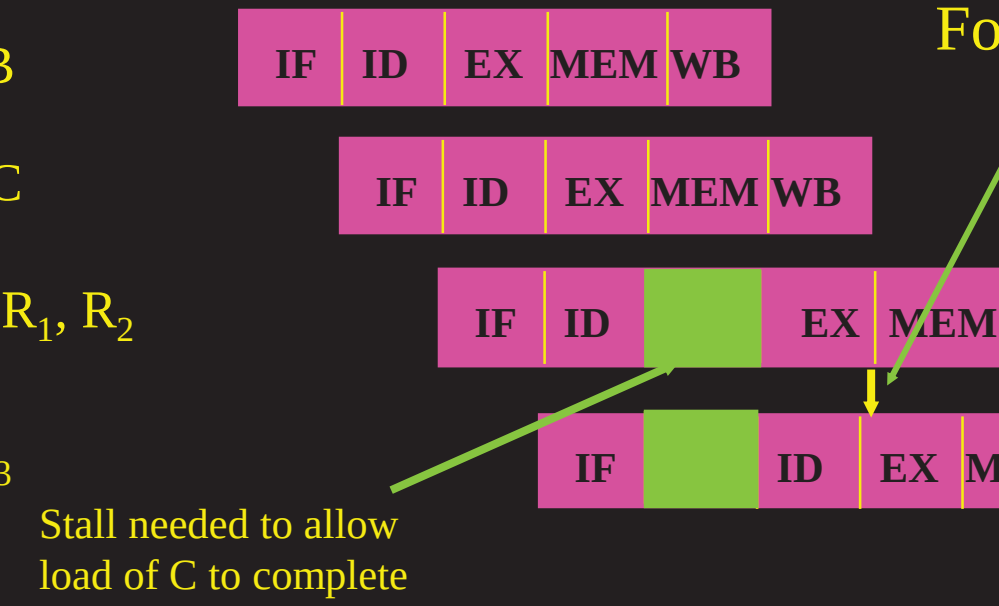


Computation Gap

- Pipeline Systems
  - Data Hazard — Stalling
    - The pipeline interlock detects a hazard and stall pipeline until the hazard is cleared .
    - This delay cycle — bubble or pipeline stall, al the load data to be generated at the time it is ne by the instruction.

Computation Gap

- Pipeline Systems
  - Data Hazard — Stalling
    - Let us look at $A \leftarrow B + C$



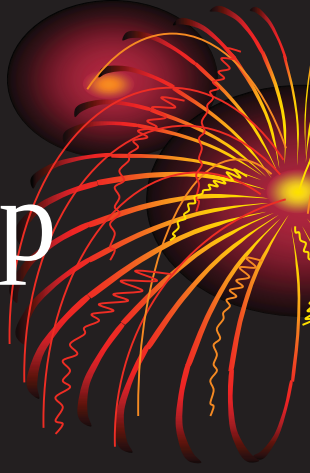
Computation Gap

- Pipeline Systems
  - Data Hazard — Example
    - Assume 30% of the instructions are load and the time the instruction following a load instru depends on the result of the load. If the ha creates a single-cycle delay, how much faster i ideal pipelined machine?

$$CPI_{ideal} = 1$$

$$CPI_{new} = (.7 * 1 + .3 * 1.5) = 1.15$$

Computation Gap



- Pipeline Systems
  - Data Hazard — Pipeline Scheduling or Instru Scheduling
    - Compiler attempts to schedule the pipeline to a the stalls by rearranging the code sequenc eliminate the hazard — Software support to a data hazard.
    - Sometimes if compiler can not schedule interlocks, a no-op instruction may be inserted.

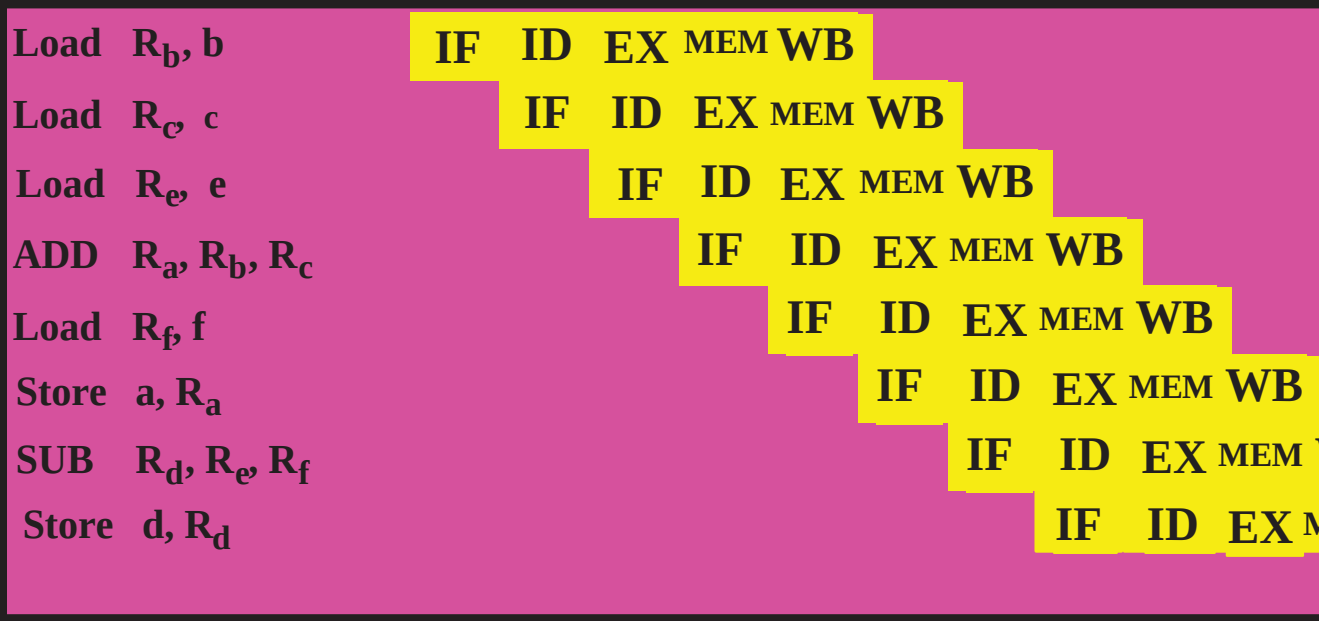
Computation Gap

- Pipeline Systems
  - Data Hazard — Pipeline Scheduling or Instru Scheduling
    - Let us look at the following sequence instructions:

```
a = b + c
d = e - f
```

Computation Gap

- Pipeline Systems
  - Data Hazard — Pipeline Scheduling or Instru Scheduling

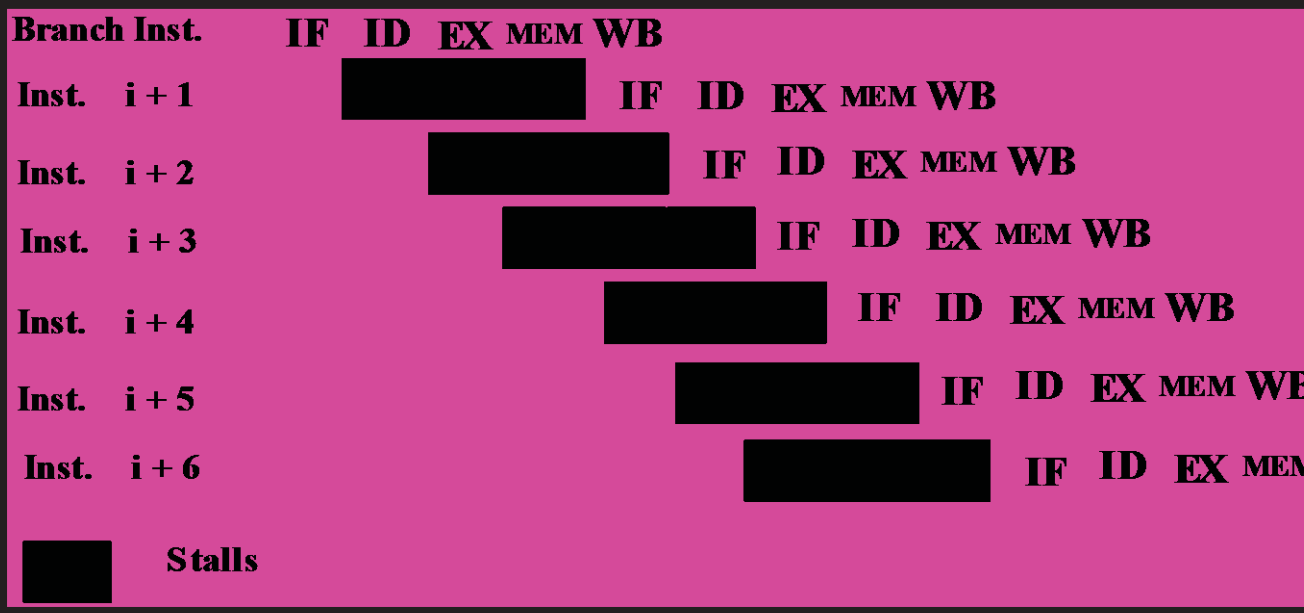


Computation Gap

- Pipeline Systems
  - Control Hazard
    - If instruction *i* is a successful branch, then the *P*
      changed at the end of *MEM* phase. This m
      stalling the next instructions for three clock cycl

Computation Gap

- Pipeline Systems
  - Control Hazard

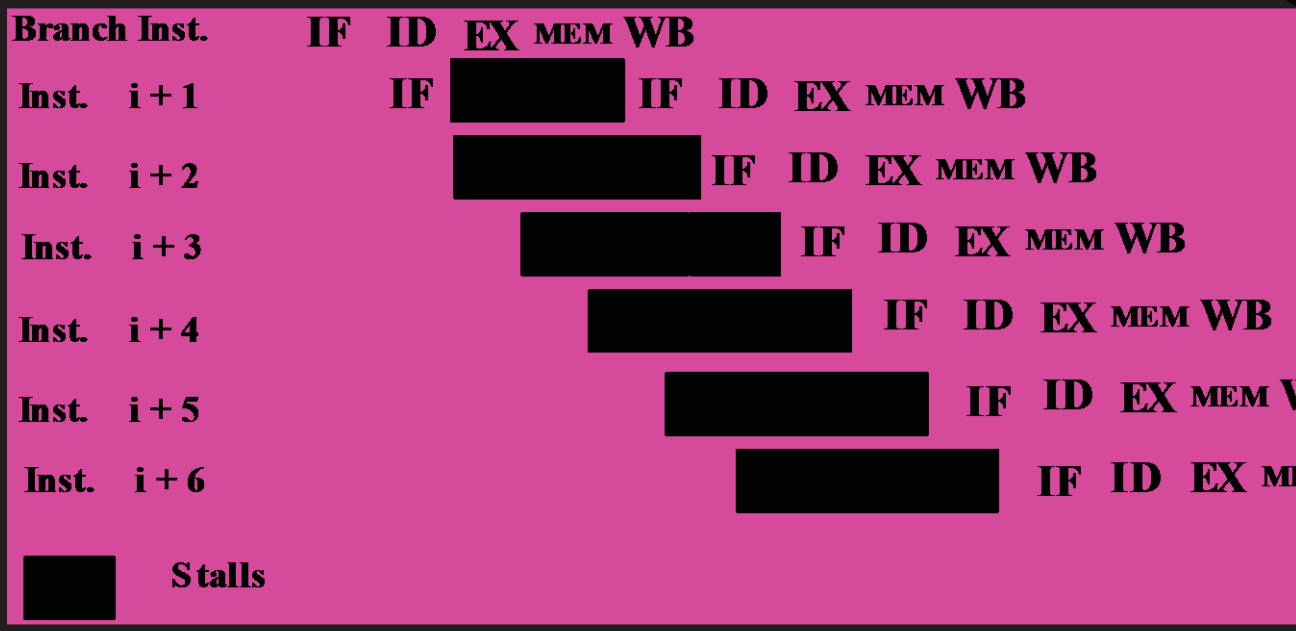


Computation Gap

- Pipeline Systems
  - Control Hazard — Observations
    - Three clock cycles are wasted for every branch.
    - However, the above sequence is not even poss since we do not know the nature of the instru until after the instruction *i + 1 is* fetched.

Computation Gap

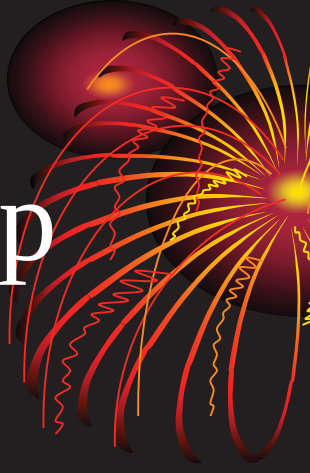
- Pipeline Systems
  - Control Hazard — Solution



Computation Gap

- Pipeline Systems
  - Control Hazard — Solution
    - Still the performance penalty is severe.
    - What are the solution(s) to speed up the pipeline

Computation Gap

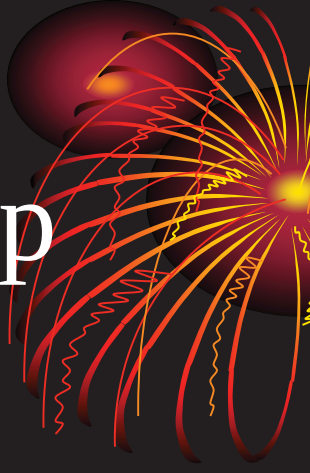


- Pipeline Systems
  - Control Hazard — Reducing pipeline br penalties
    - Detect, earlier in the pipeline, whether or no branch is successful,
    - For a successful branch, calculate the value o *PC* earlier,
    - It should be noted that, these solutions come a expense of extra hardware,

Computation Gap

- Pipeline Systems
  - Control Hazard — Reducing pipeline br penalties
    - Freeze the pipeline — Holding any instruction the branch until the branch destination is know Easy to enforce,
    - Assume unsuccessful branch — Continue to instructions as if the branch were a no instruction. If a branch is taken, then stop pipeline and restart the fetch,

Computation Gap



- Pipeline Systems
  - Control Hazard — Reducing pipeline br penalties
    - Assume the branch is successful — as soon a target address is calculated, fetch and exe instructions at the target,
    - Delayed Branch — Software attempts to mak successor instruction valid and useful.

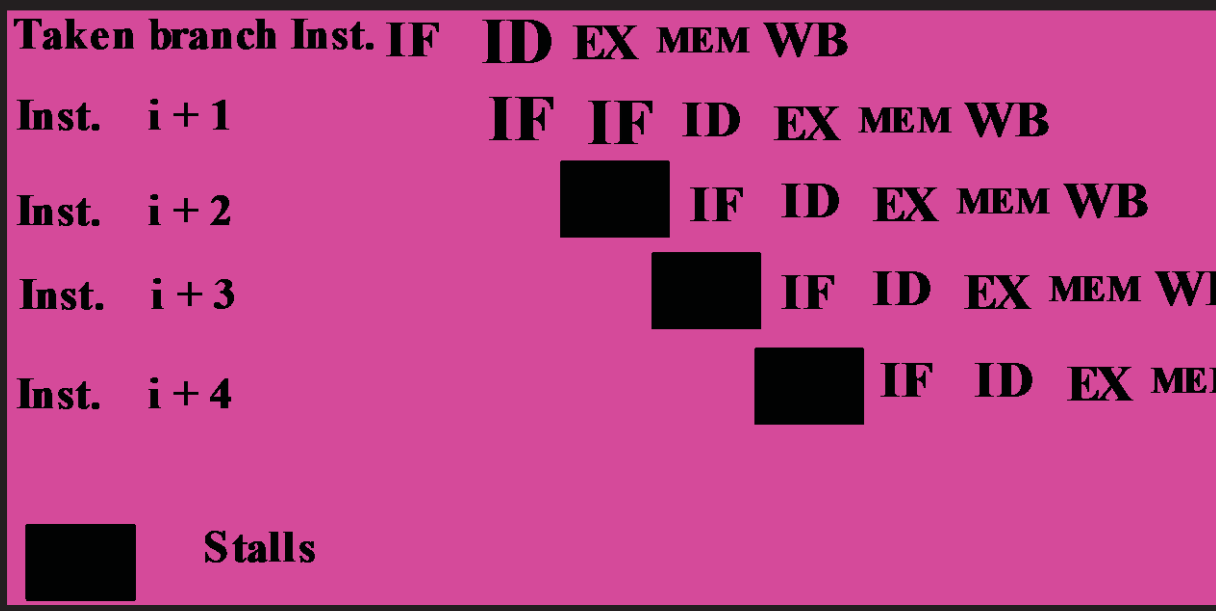
Computation Gap

- Pipeline Systems
  - Control Hazard — Reducing pipeline br penalties
    - Assume branch is not successful:

Untaken branch Inst.	IF	ID	EX	MEM	WB					
Inst. i + 1		IF	ID	EX	MEM	WB				
Inst. i + 2			IF	ID	EX	MEM	WB			
Inst. i + 3				IF	ID	EX	MEM	W		
Inst. i + 4					IF	ID	EX	ME		

Computation Gap

- Pipeline Systems
  - Control Hazard — Reducing pipeline br penalties



Computation Gap

- Pipeline Systems
  - Structural Hazard
    - For Statically Configured pipelines, one c predict precisely when a structural hazard m occur and hence it is possible to schedule pipeline so that the collisions do not occur.

Computation Gap

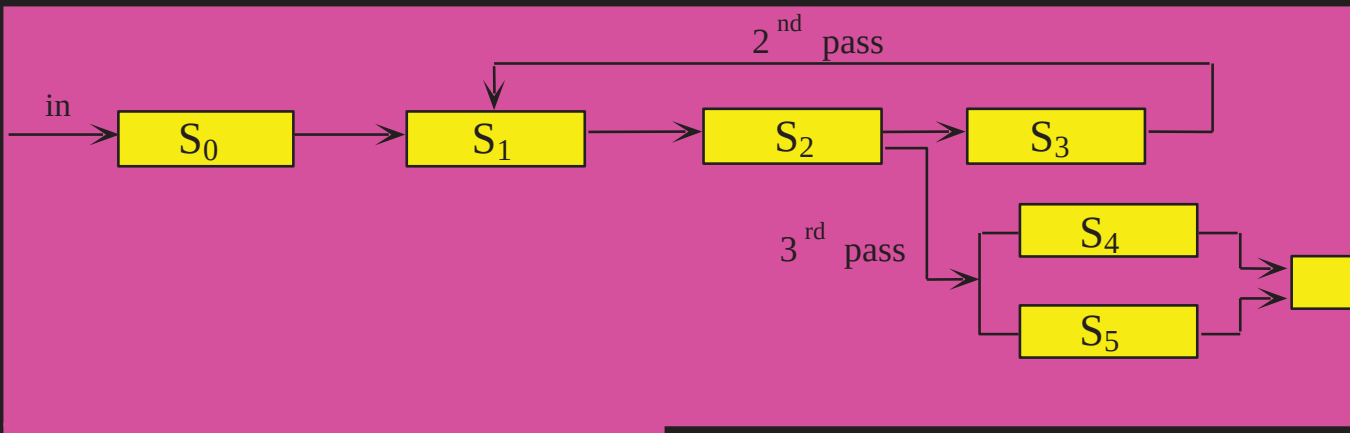
- Pipeline Systems
  - Structural Hazard
    - Let S_i ($1 \leq i \leq n$) denote a stage of a pipeline
      performs a well defined subtask with a delay tim
      δ_i.
    - Define latency as the minimum time ela
      between the initiation of two processes. There
      for a linear pipeline the latency is $\text{Max}(\delta_i)$ $1 \leq i$

Computation Gap

- Pipeline Systems
  - Structural Hazard
    - Reservation Table represents the flow of through the pipeline for one complete evaluatio a given function.
    - It is a table which shows the activation of pipeline stages at each moment of time.

Computation Gap

- Pipeline Systems
  - Structural Hazard
    - Assume the following pipeline:



$\delta_i = \Delta t \quad 0 \le i \le 6 \quad \text{and } i \ne 3$

$\delta_3 = 2\,\Delta t$

Computation Gap

- Pipeline Systems
  - Structural Hazard
    - The following is the reservation table of aforementioned pipeline organization:

Stage \ Time	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
S_0	x												
S_1		x				x				x			
S_2			x				x				x		
S_3				x	x			x	x				
S_4												x	
S_5												x	
S_6													x

Computation Gap

- Pipeline Systems
  - Structural Hazard
    - For a given pipeline organization, one can al
      derive its unique reservation table. How
      different pipeline organizations might have the
      reservation table.

Computation Gap

- Pipeline Systems
  - Structural Hazard
    - A pipeline is statically configured if it assume same reservation table for each activation.
    - A pipeline is multiply configured if the reserv table is one from a set of reservation tables.
    - A pipeline is dynamically configured if activation does not have a predetermined reserv table.

Computation Gap

- Pipeline Systems
  - Structural Hazard
    - A collision occurs if two or more activations att to use the same pipeline segment simultaneously
    - A collision will occur if reservation tables are of by *l* time units and activation of the same pip segment overlaps.
    - *l* is called a forbidden latency — two activa should not be initiated *l* time units apart.

Computation Gap

- Pipeline Systems
  - Structural Hazard
    - Given a pipeline system, one can define forbidden list *L* as the set of forbidden latencies

$$L = (l_1, l_2, ..., l_n)$$

    - For a given *L* we can define the collision vector *C* as:

$$C = (C_n, C_{n-1}, ..., C_1)$$

$$n = \text{Max}(l_j) \quad l_j \in L \text{ and}$$
$$C_i = 1 \quad \text{if } i \in L$$
$$C_i = 0 \quad \text{otherwise}$$

Computation Gap

- Pipeline Systems
  - Structural Hazard
    - The collision vector can be interpreted tha initiation is allowed at every time unit such that 0. This allows us to build a finite state diagra possible initiations.
    - Initial state is the collision vector, and each sta represented by a combination of collision ve which have led to such a state.

Computation Gap

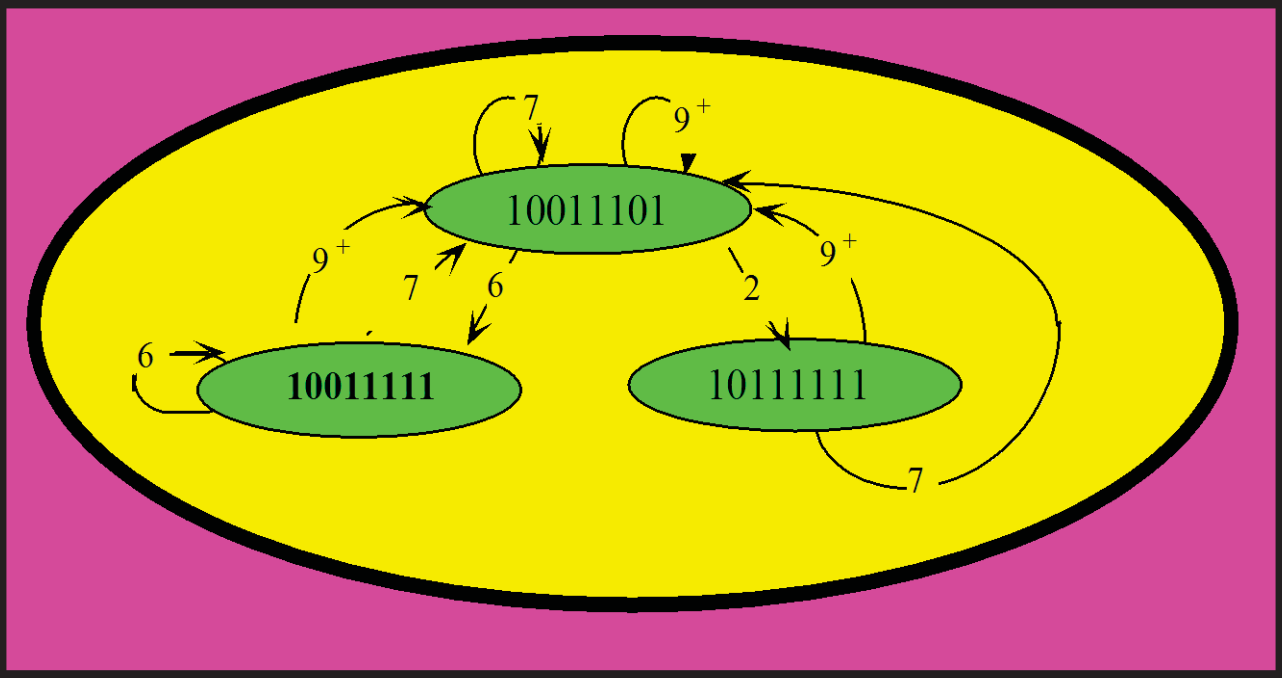
- Pipeline Systems
  - Structural Hazard
    - In case of our example we have:

$$L = (4,8,1,3,5)$$
$$C = 10011101$$

    - Initial state 10011101 indicates that we can have initiation at time 2, 6, or 7. If we have a new init at time 2 then the finite state machine would be i state $(00100111) \cup (10011101) = 10111111$. Following such a procedure then we have:

Computation Gap

- Pipeline Systems
  - Structural Hazard



Computation Gap

- Pipeline Systems
  - Structural Hazard
    - From State Diagram then one can design a control regulate the initiation of the activations.
    - In a state diagram:
      - The simple cycle is a cycle in which each state ap only once.
      - The average latency of a cycle is the sum of its late divided by the number of states in the cycle.
      - The greedy cycle is a cycle that always minimize latency between the current initiation and the ver initiation.

Computation Gap

Pipeline Systems — Example

- For a 5-stage pipeline characterized by the follo reservation table:

Stage \ Time	0	1	2	3	4	5	6	7	8
1	x								x
2		x	x					x	
3				x					
4					x	x			
5							x	x	

a) Determine forbidden list and collision vector.

b) Draw the state diagram and determine the minimal average latency and the maximum throughput.

Computation Gap

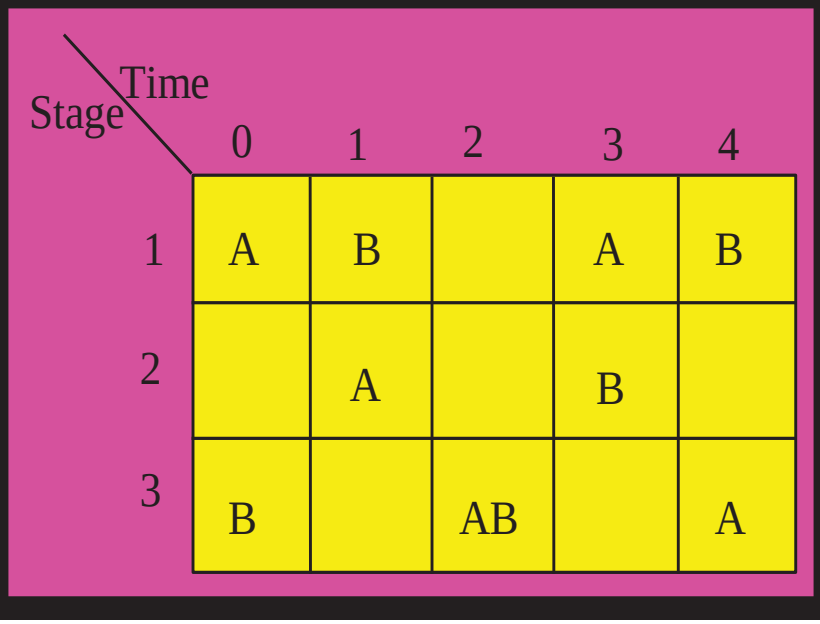
- Pipeline Systems
  - Multifunction Pipeline
    - The scheduling method for static uni-fun pipelines can be generalized for multifun pipelines.
    - A pipeline that can perform *P* distinct functions be classified by *P* overlaid reservation tables.

Computation Gap

- Pipeline Systems
  - Multifunction Pipeline
    - Each task to be initiated can be associated w function tag to identify the reservation table t used.
    - In this case collision may occur between tw more tasks with the same function tag or distinct function tags.

Computation Gap

- Pipeline Systems
  - Multifunction Pipeline
    - For example, the following reservation characterizes a 3-stage 2-function pipeline



Computation Gap

- Pipeline Systems
  - Multifunction Pipeline
    - A forbidden set of latencies for a multifunction pip is the collection of collision-causing latencies.
    - A cross-collision vector marks the forbidden late between the functions - i.e., v_{AB} represents the forb latencies between *A* and *B*. Therefore, for a *P* fun pipeline one can define P^2 cross-collision vectors.
    - P^2 cross-collision vectors can be represented b collision matrices.

Computation Gap

- Pipeline Systems
  - Multifunction Pipeline
    - For our example we have:

Cross Collision Vectors

v_{AA}=(0110) v_{AB}=(1011)

v_{BA}=(1010) v_{BB}=(0110)

Collision Matrices

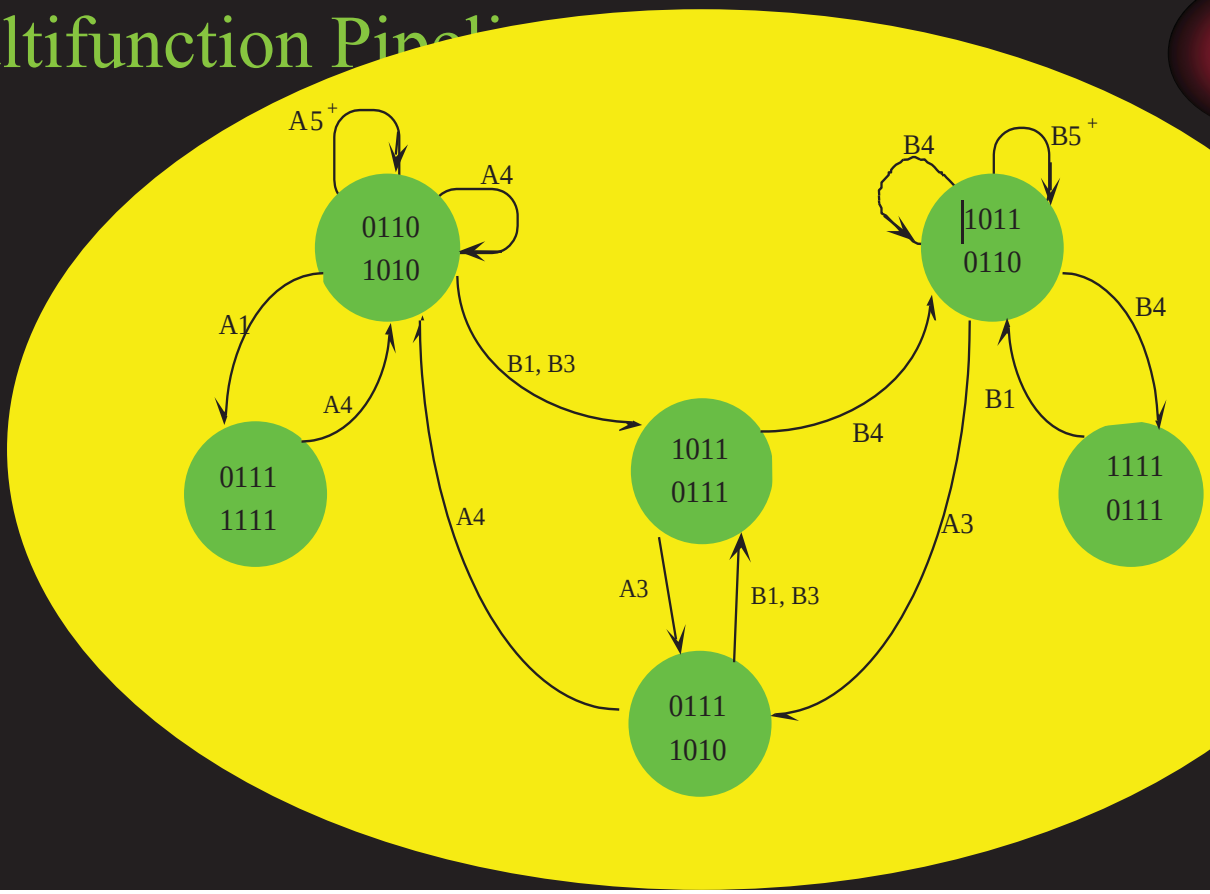
$$M_A = \begin{bmatrix} 0110\ (AA) \\ 1010\ (BA) \end{bmatrix} \qquad M_B = \begin{bmatrix} 1011\ (AB) \\ 0110\ (BB) \end{bmatrix}$$

Computation Gap

- Pipeline Systems
  - Multifunction Pipeline
    - Similar to the uni-function pipeline, one can us collision matrices in order to construct a diagram.

Computation Gap

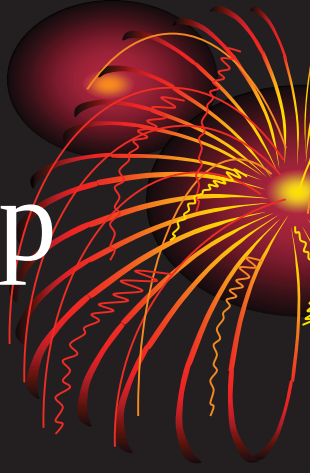
- Pipeline Systems
  - Multifunction Pipeli



Computation Gap

- Pipeline Systems
  - Vector Processors
    - A vector processor is equipped with multiple v pipelines that can be concurrently used u hardware or firmware control.
    - There are two groups of pipeline processors:
      - Memory-to-Memory Architecture
      - Register-to-Register Architecture

Computation Gap



- Pipeline Systems
  - Vector Processors
    - Memory-to-Memory architecture that support pipeline flow of vector operands directly from memory to pipelines and then back to the mem (Cyber 205).
    - Register-to-Register architecture that uses v registers as operands for the functional pipe (Cray series that use size registers and Fujitsu 2000 series that use reconfigurable vector regist

Computation Gap

- Pipeline Systems
  - Vector Processors
    - Vector machines allow efficient use of pipeli
      while reducing memory latency and pip
      scheduling penalties.
    - Computations on vector elements are m
      independent from each other — lack of
      hazards.

Computation Gap

- Pipeline Systems
  - Vector Processors
    - A vector instruction is equivalent to a loop, implies:
      - Smaller program size, hence reducing the instr bandwidth requirement.
      - Fewer number of control hazards.
    - Vector instructions initiate regular operand pattern — allowing efficient use of mer interleaving and efficient address calculations.

Computation Gap

- Pipeline Systems
  - Vector Processors — Vector Stride
    - What if adjacent vector elements of a vector oper are not positioned in sequence in the memory.
    - The distance separating elements that ought to be m into a single vector is called the stride.
    - Almost all vector machines allow access to vectors any constant stride. Some constant strides may memory-bank conflict.

Computation Gap

- Pipeline Systems
  - Vector Processors — Chaining
    - Chaining allows a vector operation to start as as the individual elements of its vector so operand become available.
    - Result of the first functional unit (pipeline) in chain are forwarded to the second functional uni

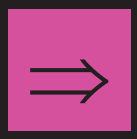
Computation Gap

- Pipeline Systems
  - Efficient Use of Vector Processors — Mem to-Memory Organization
    - Increase the vector size — if possible:
      - Change the nesting order of the loop,
      - Convert multidimensional arrays into one-dimen arrays,
      - Rearrange data into unconventional forms so that sm vectors may be combined into a single large vector.
    - Perform as many operations on an input vecto possible before storing the result vector back in main memory.

Computation Gap

- Pipeline Systems
  - Efficient Use of Vector Processors — Mem
    to-Memory Organization
    - Changing the nesting order of the loop:

```
Do      I = 1, 100
        A(I, 1:60) = 0
End
```



```
Do      J = 1, 60
        A(1:100,J) =
End
```

Computation Gap

- Pipeline Systems
  - Efficient Use of Vector Processors — Mem to-Memory Organization
    - Convert multidimensional arrays into dimensional arrays:

```
Do      I = 1, 100
        A(I, 1:60) = 0
End
```

$\Rightarrow$

```
A(1:6000) = 0
```

Computation Gap

- Pipeline Systems
  - Efficient Use of Vector Processors — Reg to-Register Organization
    - Values often used in a program should be kept in int registers.
    - Perform as many operations on an input vecto possible before storing the result vector back in the memory.
    - Organize vectors into sections of size equal to the l of the vector registers — Strip mining.
    - Convert multidimensional arrays into one-dimens arrays.

Computation Gap

- Question
  - Compare and contrast Memory-to-Memory Register-to-Register pipeline systems aga each other.