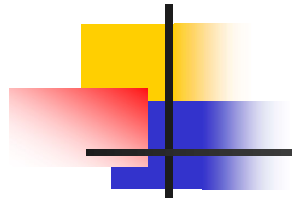


Unit-5

Objectives

- Multithreading
- Need for Multicore Architectures
- Types of Multiprocessors



Multithreading



Multithreading

- Thread : is a process with its own instructions and data
- It may be apart of a parallel program or represent an independent program on its own.
- Multithreading: is the execution of multiple threads simultaneously.
- ILP exploits implicit parallelism while TLP exploits explicit parallelism



Multithreading

- Multiple threads to share the functional units of a single processor in an overlapping fashion
- processor must duplicate the resources
 - Separate registers
 - PC
 - Page table
 - Memory is shared thru virtual memory mech.
 - H/W must support thread switching



Multithreading Classification

- Fine-grained multithreading
- Coarse-grained multithreading
- Simultaneous Multithreading



Fine grain Multithreading

- Switches between threads on each instruction
- Execution of multiples threads to be interleaved.
- Interleaving is done in a round-robin fashion
- CPU must be able to switch threads on every clock cycle



Fine grain Multithreading

- Advantage:
 - it can hide the throughput losses that arise from both short and long stalls.
- Disadvantage:
 - it slows down the execution of the individual threads.



Coarse-grained multithreading

- Switches threads only on costly stalls
 - Ex: level two cache misses
- Alternative to fine grained multithreading
- CPU with coarse-grained multithreading issues instructions from a single thread
- Advantage:
- Is less likely to slow the processor down .
 - since instructions from same thread will only be issued, when a thread encounters a costly stall



Coarse-grained multithreading

- Drawback:
 - limited in its ability to overcome throughput losses
 - especially from shorter stalls
- when a stall occurs, the pipeline must be emptied or frozen



Simultaneous Multithreading

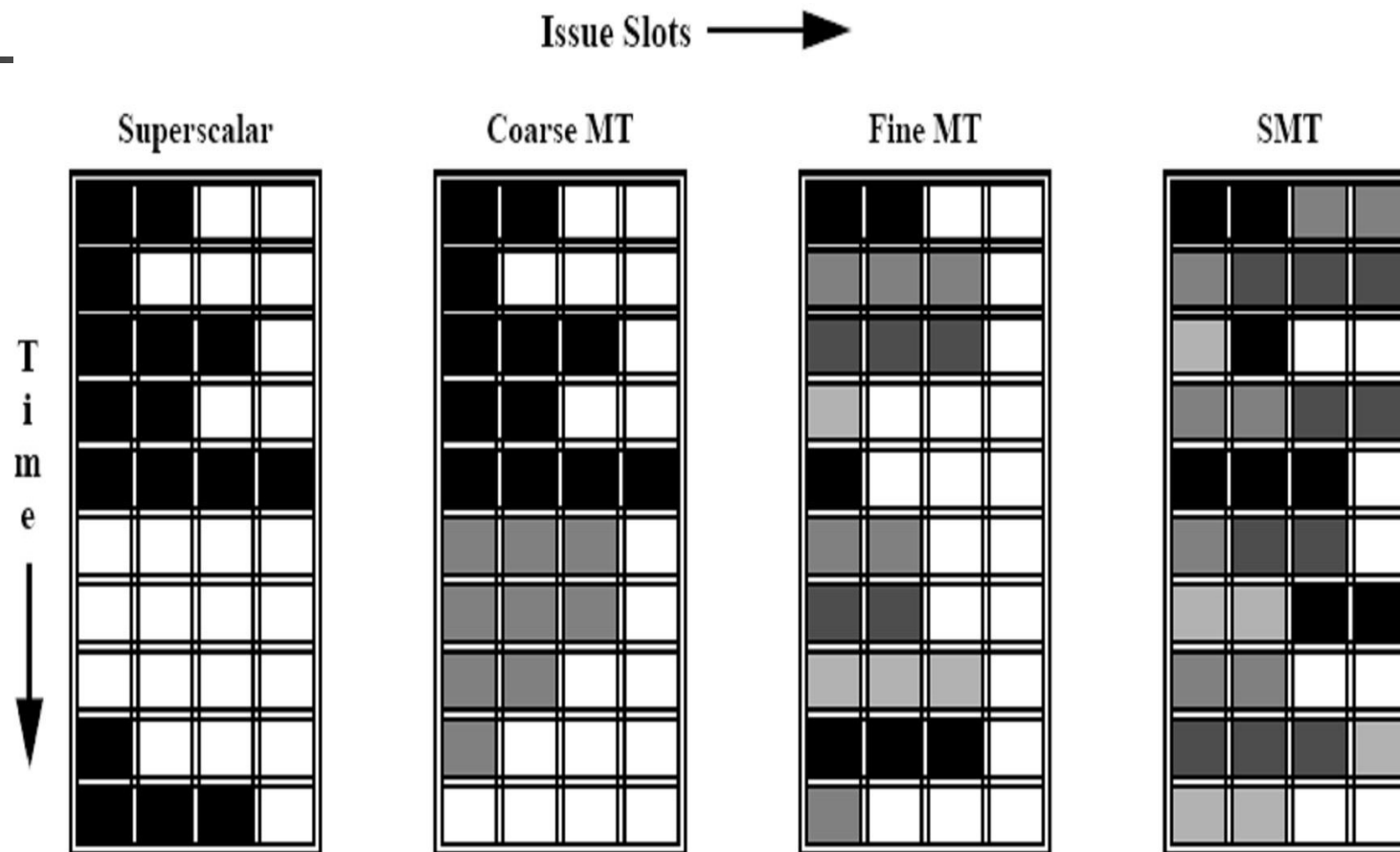
- It exploit TLP at the same time it exploits ILP
- SMT is multiple-issue processors often have more functional unit parallelism available
- SMT uses the concepts like
 - Multiple-issue
 - Register Renaming
 - Data forwarding
 - Static scheduling
 - Dynamic scheduling



Example

- superscalar with no multithreading support
- superscalar with coarse-grained multithreading
- superscalar with fine-grained multithreading
- superscalar with simultaneous multithreading.

Example





Example

- Horizontal dimension represents the instruction issue capability in each clock cycle.
- The vertical dimension represents a sequence of clock cycles.
- Empty box indicates that the corresponding issue slot is unused in that clock cycle



Example

- Superscalar without MT:
 - Exploits ILP
 - No Multithreading facility
 - Large no of processor idle cycles.
- Coarse Grain MT:
 - In the coarse-grained multithreading the long stalls are partially hidden by switching to another thread
 - since thread switching only occurs when there is a stall there are likely to be some fully idle cycles



Example

- fine-grained MT:
 - the interleaving of threads eliminates fully empty Slots
 - only one thread issues instructions in a given clock cycle
 - ILP limitations still lead to a significant number of idle slots within individual clock cycles.
 - SMT
 - ILP and TLP are exploited
 - multiple threads using the issue slots in a single clock cycle
 - No issue slot is idle



Multithreading

- Advantages:
 - If a thread gets lot of cache misses then the other threads can continue by using the computing resources.
 - If several threads work on the same set of data then better cache usage and sync can be achieved
 - If a thread can not use all the computing resources running other threads permit to use these resources.



Multithreading

- Disadvantages:
 - Multiple threads can interfere with each other when sharing h/w resources like cache ,TLB.
 - H/W support for Multithreading is more visible to S/W.
- Applications:
 - Used in server side applications



Limitations of Single core ...



- Smaller transistors = faster processors.
- Faster processors = increased power consumption.
- Increased power consumption = increased heat.
- Increased heat = unreliable processors.

Limitations of Single core...

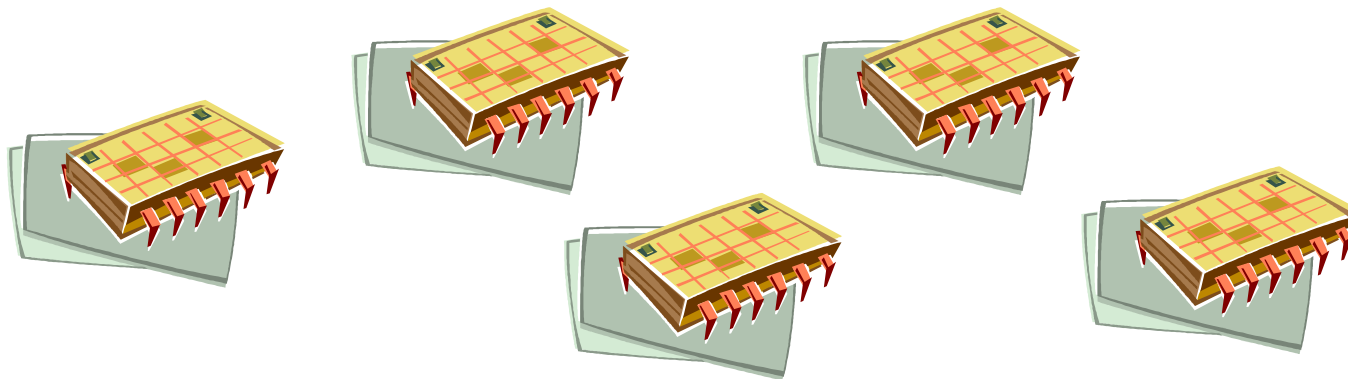


- A simple Thump rule is that
 - For every 1% rise in the clock frequency you will see 3% rise in the power consumption
 - Thus the heat dissipation also increases.



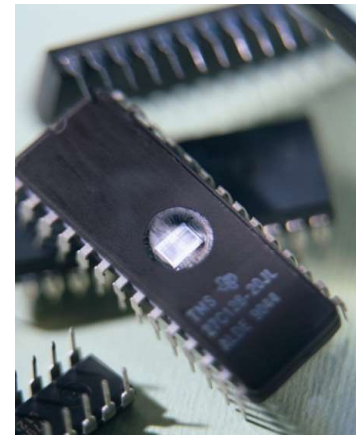
An intelligent solution...

- Instead of designing and building faster microprocessors, put multiple processors on a single integrated circuit.



An intelligent solution

- Move away from single-core systems to multicore processors.
- “core” = central processing unit (CPU)
- Introducing parallelism!!!





Why Multicores ?

- Difficult to make single core clock frequency higher
- Many new applications are multithreaded
- General trend in Computer Architecture is shift toward more parallelism



Multicore Architectures ...

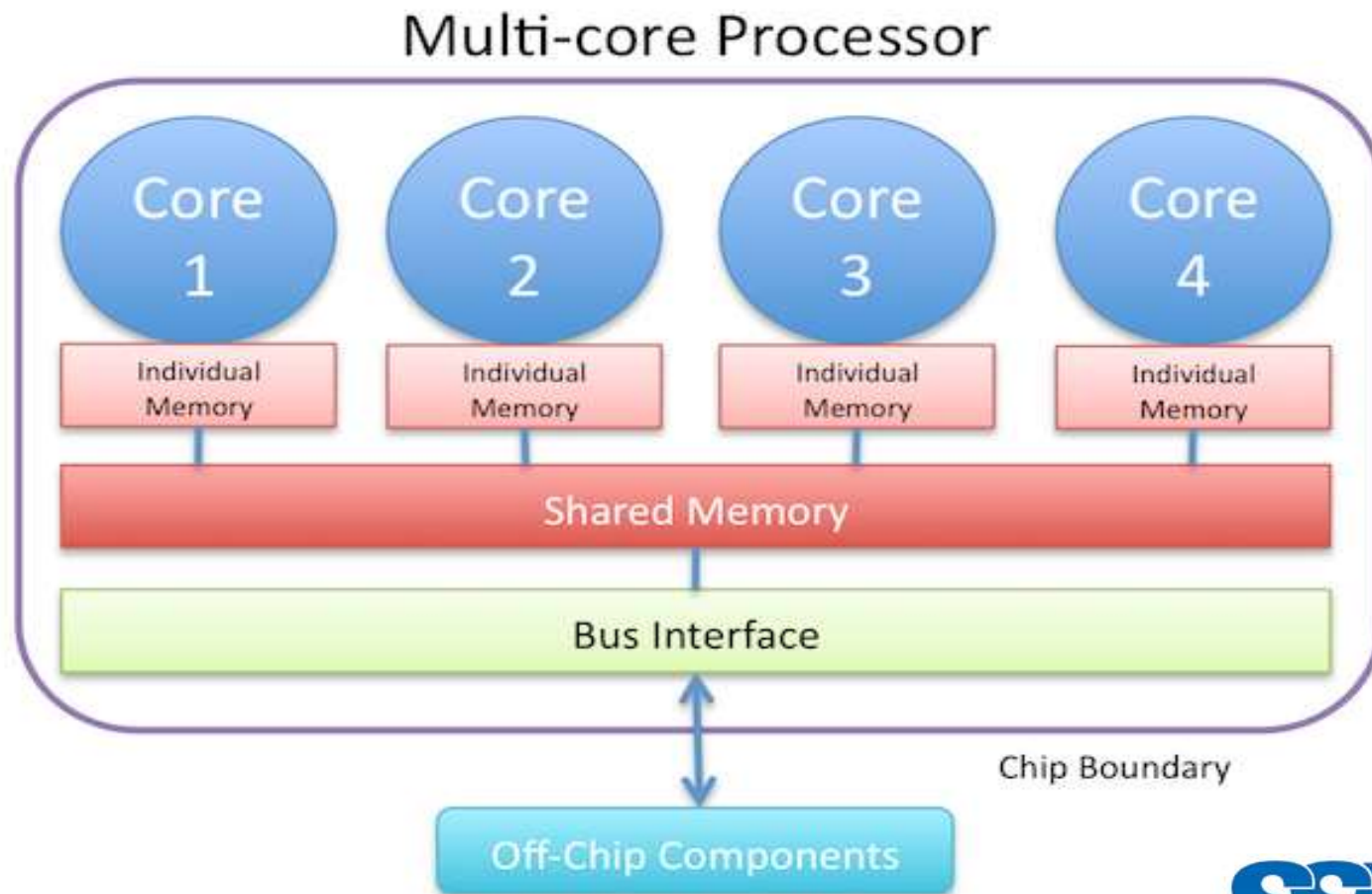
- Multi-core is a design in which a single physical processor contains the core logic of more than one processor.
- It's a special kind of Multiprocessor.
- All processors are on the same chip



Multicore Architectures...

- Multicore processors are **MIMD**
 - Different cores executes different threads (Multiple Instructions) ,operates on different parts of memory(Multiple Data)
 - Multicore is a Shared Memory Multiprocessors. All cores share the same memory.

Multicore Architectures...





Multicore Architectures...

- contain two or more distinct cores in the same physical package
- each core has its own execution pipeline
- each core has the resources required to run without blocking resources needed by the other software threads.
- core design enables two or more cores to run at somewhat slower speeds and at much lower temperatures



Multicore Architectures

- combined throughput of these cores delivers processing power greater than the maximum available today on single-core processors and at a much lower level of power consumption
- Ex: 16 core MIT RAW processor operates at 425 MHz can perform 100 time the number of operations per second than Intel Pentium-3 with 600MHz.

Advantages



- Occupies less space on PCB
- Higher throughput
- Consume less power
- Cache coherency can be greatly improved
- Performs more operations/sec with less frequency

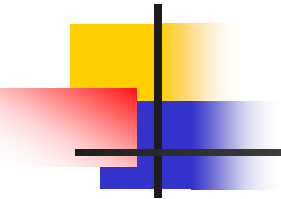


Disadvantages

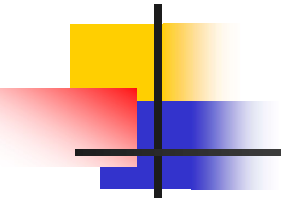


- Maximizing the utilization of the computing resources provided by multi-core processors requires adjustments both to the operating system (OS) support and to existing application software
- They are more difficult to manage thermally than lower-density single-chip designs

Multicore applications

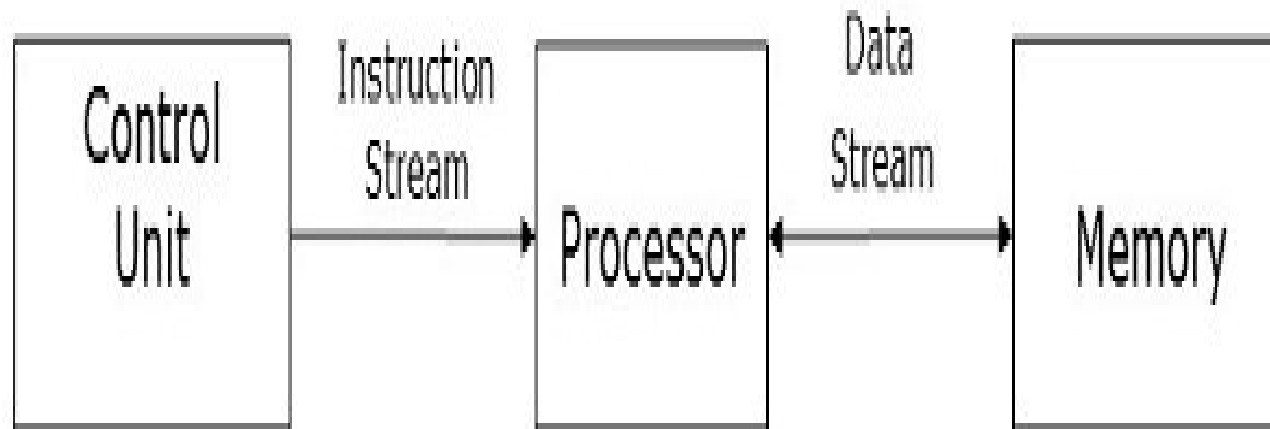
- 
- Data base servers
 - Web servers
 - Compilers
 - Multimedia Applications
 - Scientific Applications
 - General applications with TLP as opposed to ILP
 - Downloading s/w while running Anti virus s/w
 - Editing photo while recording TV show.

Flynn's Classification of Multiprocessors



<i>classic von Neumann</i>	
SISD Single instruction stream Single data stream	(SIMD) Single instruction stream Multiple data stream
MISD Multiple instruction stream Single data stream <i>not covered</i>	(MIMD) Multiple instruction stream Multiple data stream

SISD

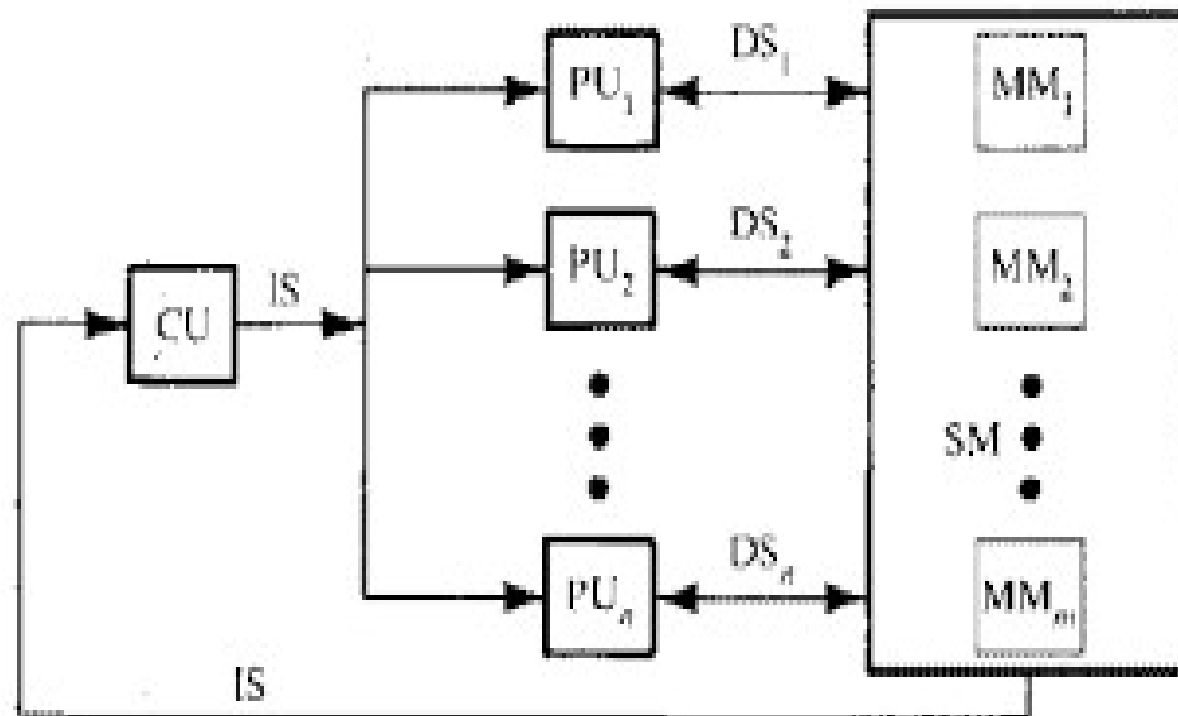




SISD

- Single instruction stream, single data stream (SISD)
 - Executes one instruction at a time and operates on single data
 - Fu's are pipelined
 - Von Neumann Architecture
 - Ex: IBM 360, IBM 701 etc

SIMD

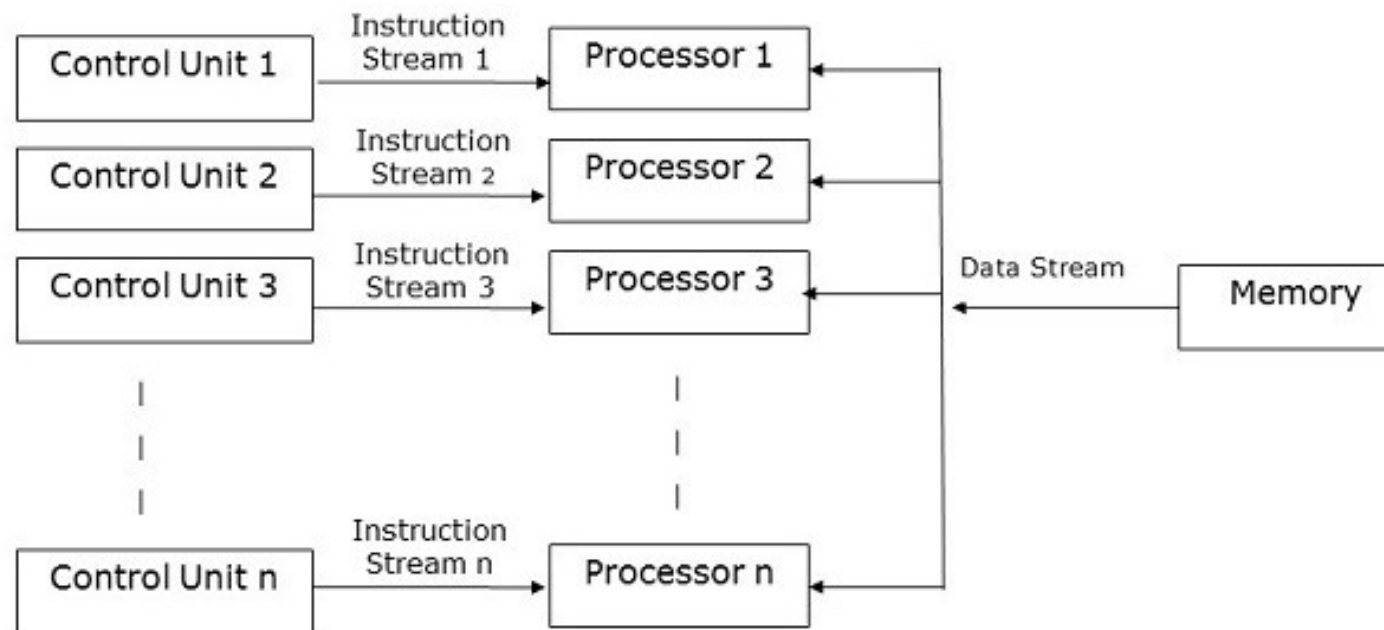




SIMD

- Single instruction stream, multiple data streams (SIMD)
- CU broadcast single instruction to all the Fus and operates on different data.
 - Vector architectures
 - Multimedia extensions
 - Graphics processor units
- Ex: TI ASC, BBN TC 2000

MISD



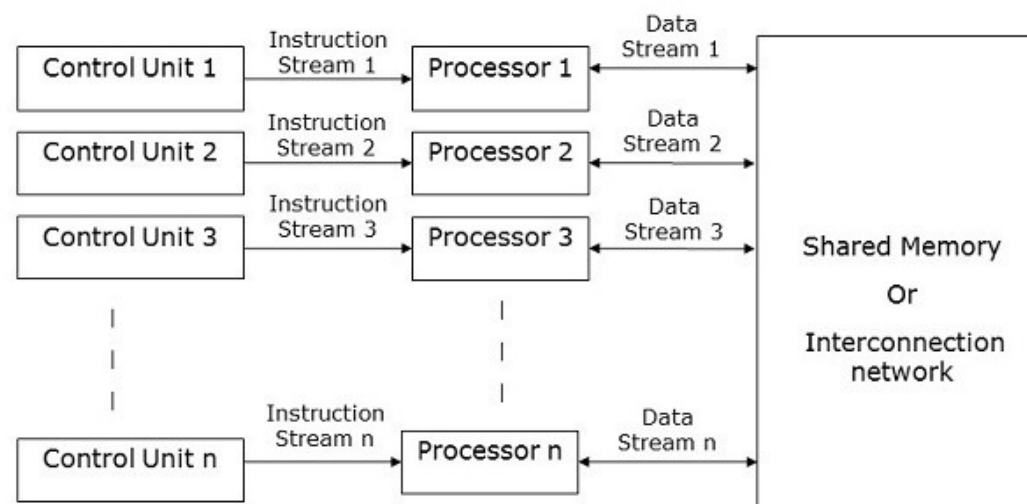


MISD

- Multiple instruction streams, single data stream (MISD)
 - Executes different instructions but operates on single data stream
 - No commercial implementation
 - Ex: Systolic Arrays



MIMD





MIMD

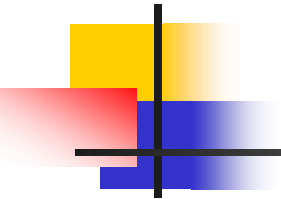
- Multiple instruction streams, multiple data streams (MIMD)
 - Different instructions will be executed with different data
 - Multiple SISD
 - Tightly-coupled MIMD (Shared Memory Multiprocessor)
 - Loosely-coupled MIMD (Distributed Memory Multiprocessor)
- Ex: IBM 3090, Cray x-MP, Cray y-MP



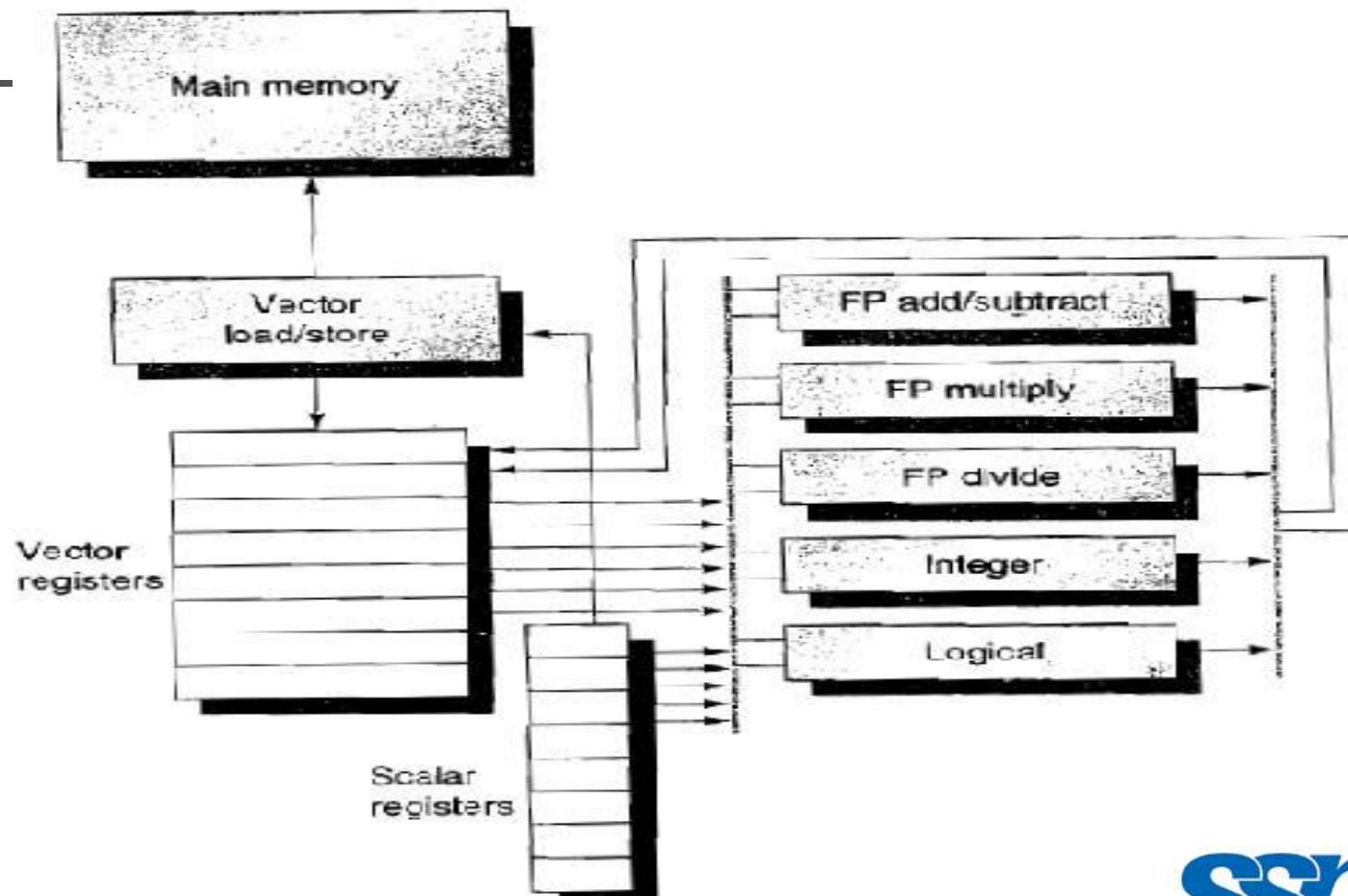
Vector Processor

- Efficient way to execute a vectorizable application is by Vector processor- Jim Smith
- **vector processor** is a CPU that executes instructions that operate on arrays of data.
 - It collects set of data elements put them in the register file
 - operates on the data in those register files and stores the results back in memory.
 - These reg files acts like buffers and hide the memory latency.

Vector processor

- 
- SIMD classification
 - Also be called as **array processor**
 - Improves performance on numerical simulations
 - Used in Video game console and Graphic accelerators
 - Ex: VIS, MMX, SSE, AltiVec and AVX

VMIPS



VMIPS

- It is loosely based on cray-1
- VMIPS instruction set
 - Scalar portion is similar to MIPS
 - Vector portion is logical vector extension of MIPS
- Registers:
 - It has 8 vector registers
 - Fixed length holding
 - Each vector register holds single vector
 - Each vector register holds 64 elements of 64 bits



VMIPS

- Vector registers
 - Vector register file has 16 read and 8 write ports
 - Supply operands to VFUs.
 - Registers and the FUs are connected by a pair of cross bar switches (thick gray lines)
- Scalar registers
 - 32 GPRs and 32 FPRs as in MIPS.
 - These supply operands to VFUs
 - Supply addresses to L/S units.





VMIPS

- Vector functional units
 - Each unit is fully pipelined
 - start a new operation on every clock cycle
 - Control Unit is needed to detect hazards
 - structural hazards for functional units
 - data hazards on register accesses



VMIPS

- VMIPS has five functional units
 - Integer unit
 - Logical Unit
 - Floating point Add/Sub
 - Floating point Multiply
 - Floating point Divide



VMIPS

- VMIPS has scalar architecture like MIPS.
- Vector load/store unit-
 - vector L/S unit loads/stores a vector to/from memory.
 - This unit is fully pipelined
 - words can be moved b/w the vector reg and memory one word per clock cycle
 - This unit also handles scalar loads and stores.

How Vector Processors Work: An Example

- Let's take a typical vector problem $Y = a * X + Y$
- X and Y are vectors resident in memory
- a is a scalar
- This problem is the so-called SAXPY or DAXPY
 - SAXPY – single precision $a * X + Y$
 - DAXPY – double precision $a * X + Y$

How Vector Processors Work: An Example

- MIPS code for the DAXPY loop

•	L.D	FO,a	;load scalar a
•	DADDIL.D	R4,Rx,#512	;last address to load
LOOP:	LD	F2,0(Rx)	;load X[i]
•	MUL.D	F2,F2,FO	;a x X[i]
•	L.D	F4,0(Ry)	;load Y[i]
•	ADD.D	F4,F4,F2	;a x X[i] + Y[i]
•	S.D	F4,9(Ry)	;store into Y[i]
•	DADDIU	Rx,Rx,#8	;increment index to X
•	DADDIU	Ry,Ry,#8	;increment index to Y
•	DSUBU	R20,R4,Rx	;compute bound
•	BNEZ	R20,Loop	;check if done

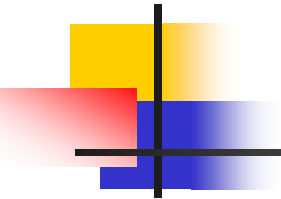


How Vector Processors Work: An Example

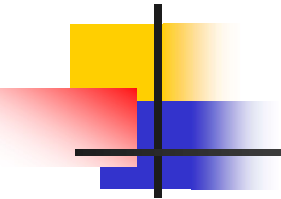
- VMIPS code for DAXPY

- LD F0,a ;load scalar a
- LV V1,R ;load vector X
- MULVS.D V2,V1,F0 ;vector-scalar multiply
- LV V3,Ry ;load vector Y
- ADDW.D V4,V2,V3 ;add
- SV V4,Ry ;store the result

How Vector Processors Work: An Example

- 
- vector processor reduces the instruction bandwidth
 - executes only 6 instructions vs almost 600 for MIPS
 - It occurs because the vector operations work on 64 elements
 - overhead instructions that constitute half the loop on MIPS are not present in the VMIPS code
 - compiler produces vector instructions for such a sequence
 - resulting code spends its time running in vector mode , the code is said to be vectorized or vectorizable.

How Vector Processors Work: An Example

- 
- Loops can be vectorized when they do not have dependences between iterations of a loop, are called loop-carried dependences.
 - Another important difference between MIPS and VMIPS is the frequency of pipeline interlocks.
 - In the MIPS code, every ADD. D must wait for a MUL. D, and every S. D must wait for the ADD. D.
 - On the vector processor, each vector instruction will only stall for the first element in each vector, and then subsequent elements will flow smoothly down the pipeline.

How Vector Processors Work: An Example

- Vector architectures call forwarding of element dependent operations chaining, in that the dependent operations are "chained" together.
- Thus, pipeline stalls are required only once per vector instruction, rather than once per vector element.



Graphics Processing Unit

- A graphics processing unit (GPU), is similar CPU
- Designed specifically for performing the complex mathematical and geometric calculations that are necessary for graphics rendering.



Graphics Processing Unit

- A graphics processing unit (GPU) is a computer chip that performs rapid mathematical calculations, primarily for the purpose of rendering images.
- occasionally called **visual processing unit (VPU)**
- GPU is able to render images more quickly than a CPU because of its parallel processing architecture
- Nvidia introduced the first GPU, the GeForce 256, in 1999
- Others include AMD, Intel and ARM.
- In 2012, Nvidia released a virtualized GPU, which offloads graphics processing from the server CPU in a virtual desktop infrastructure.





Graphics Processing Unit

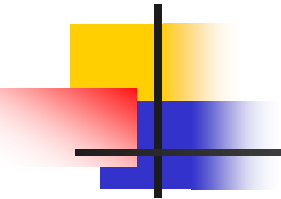
- GPUs are used in
 - Embedded Systems
 - Mobile phones
 - Personal computers
 - Workstations
 - Game consoles

GPU Vs CPU

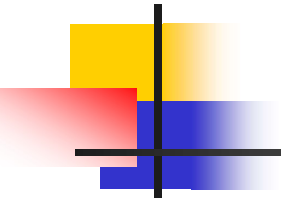
- A GPU is tailored for highly parallel operation while a CPU executes programs serially
- For this reason, GPUs have many parallel execution units and higher transistor counts, while CPUs have few execution units and higher clock speeds
- A GPU is for the most part deterministic in its operation
- GPUs have much deeper pipelines (several thousand stages vs 10-20 for CPUs)
- GPUs have significantly faster and more advanced memory interfaces as they need to shift around a lot more data than CPUs



What are GPU's Growth?

- 
- Entertainment Industry has driven the economy of these chips?
 - Males age 15-35 buy \$10B in video games / year
 - Moore's Law ++
 - Simplified design (stream processing)
 - Single-chip designs

GPU

- 
- Very Efficient For
 - Fast Parallel Floating Point Processing
 - Single Instruction Multiple Data Operations
 - High Computation per Memory Access
 - Not Efficient For
 - Double Precision
 - Logical Operations on Integer Data
 - Branching-Intensive Operations
 - Random Access, Memory-Intensive Operations



Summary

- ILP & Issues in ILP
- Multithreading
- Need for Multicore Architectures
- Types of Multiprocessors



Problems

- Suppose that a vector processor has a memory system in which it takes 10 cycles to load a simple 64-bit word from memory. How many banks are needed so that a stream of loads can on average require only one cycle per load



Problems

- A Multistage network has 3 stages. How many processors can interconnect using 2×2 switch at each stage and four switches at each stage?