Flynn’s Classification
&
Vector processor

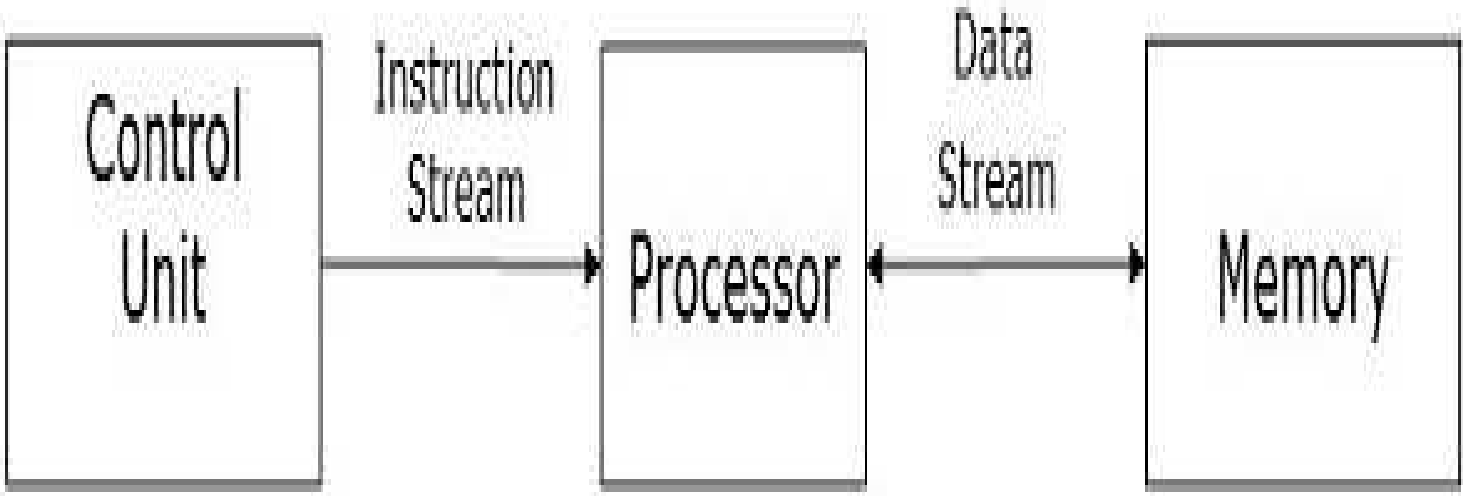
Flynn's Classification of Multiprocessors

SISD Single instruction stream Single data stream	(SIMD) Single instruction stream Multiple data stream
MISD Multiple instruction stream Single data stream	(MIMD) Multiple instruction stream Multiple data stream

classic von Neumann

not covered

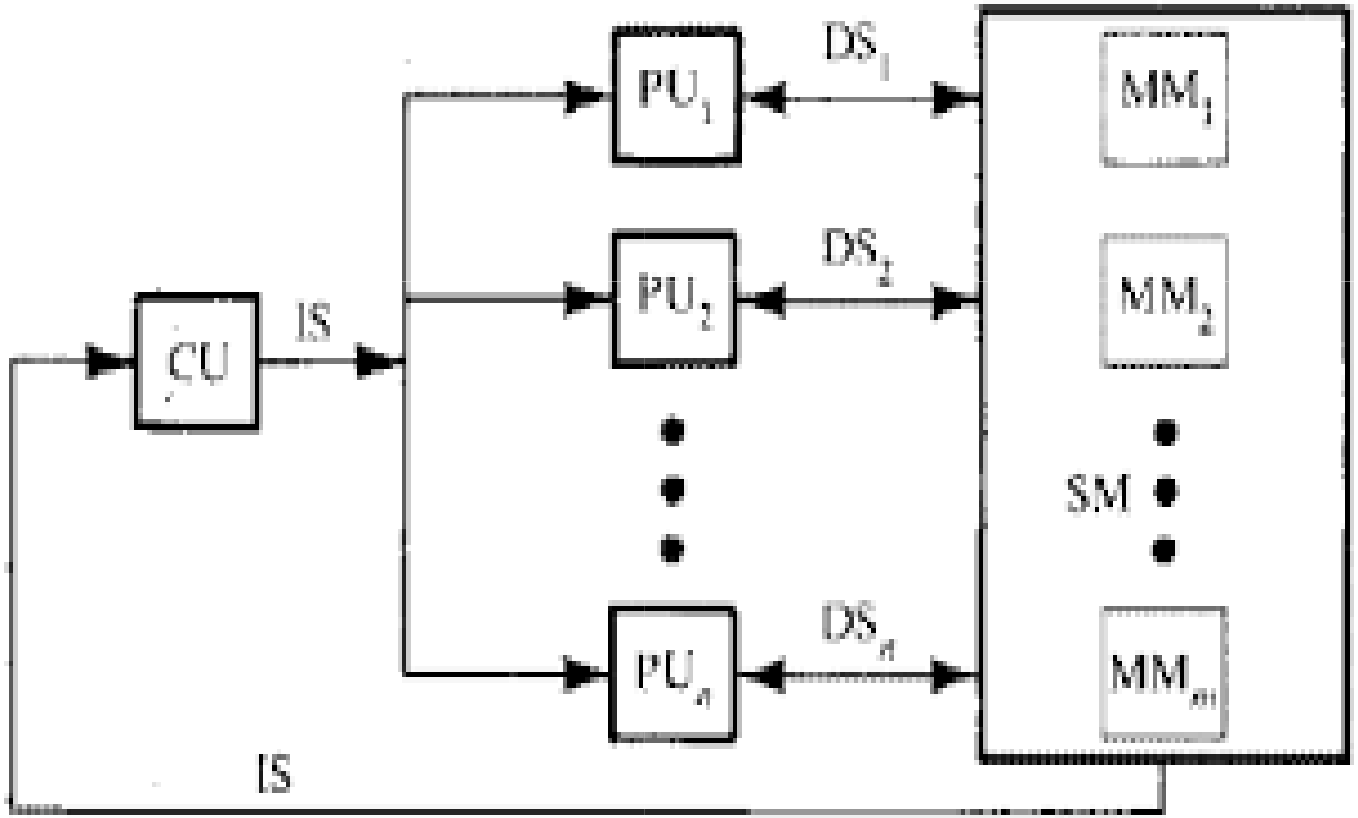
SISD


Control Unit
Instruction Stream
Processor
Data Stream
Memory

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

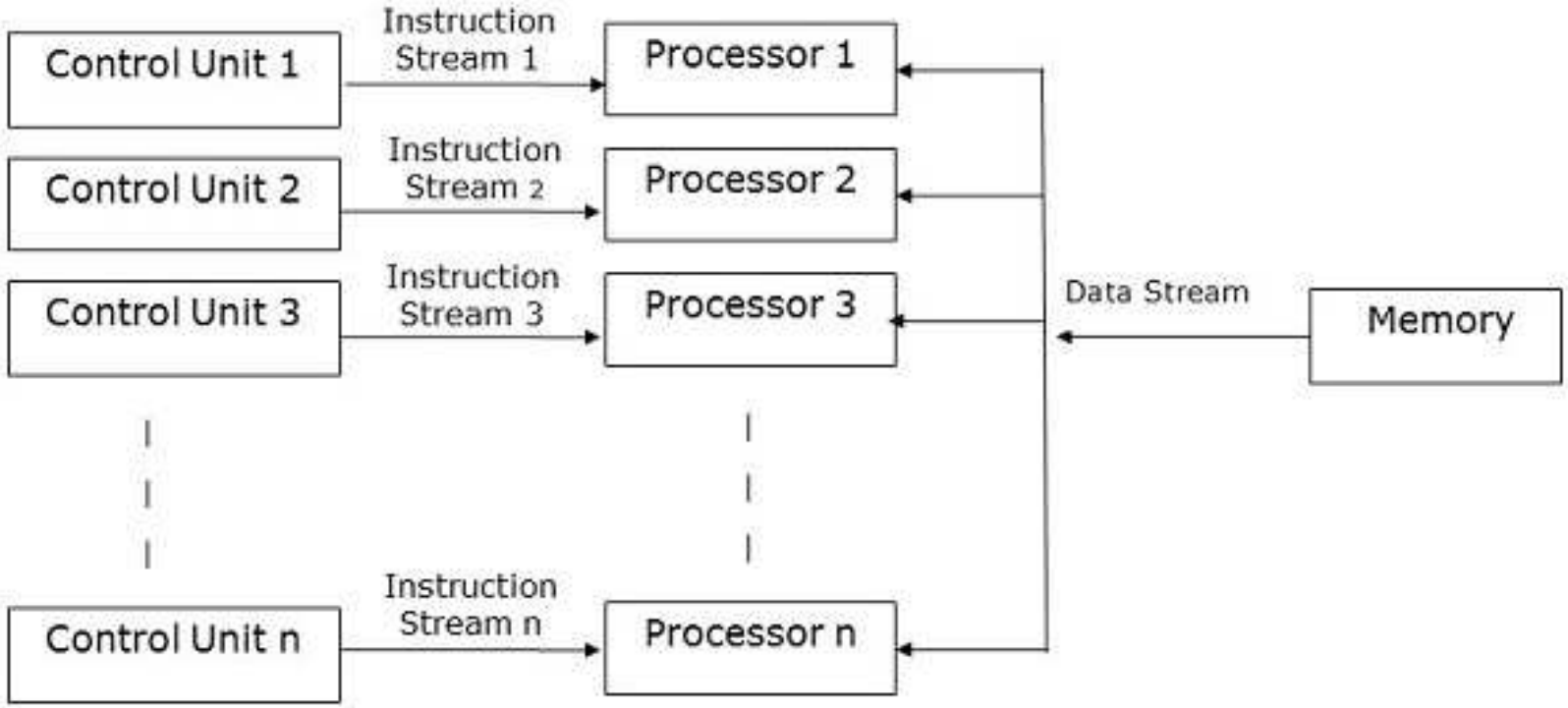
SIMD



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

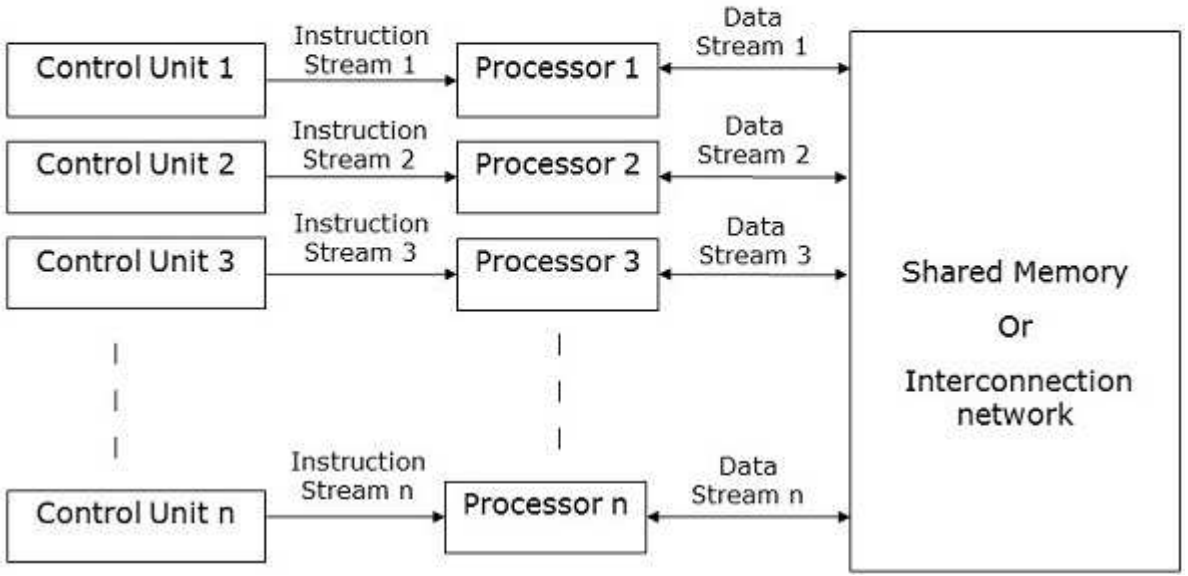
MISD


Control Unit 1
Instruction Stream 1
Processor 1
Control Unit 2
Instruction Stream 2
Processor 2
Control Unit 3
Instruction Stream 3
Processor 3
Data Stream
Memory
Control Unit n
Instruction Stream n
Processor n

MISD

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

MIMD


Control Unit 1
Instruction Stream 1
Processor 1
Data Stream 1
Control Unit 2
Instruction Stream 2
Processor 2
Data Stream 2
Control Unit 3
Instruction Stream 3
Processor 3
Data Stream 3
Shared Memory
Or
Interconnection network
Control Unit n
Instruction Stream n
Processor n
Data Stream n

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

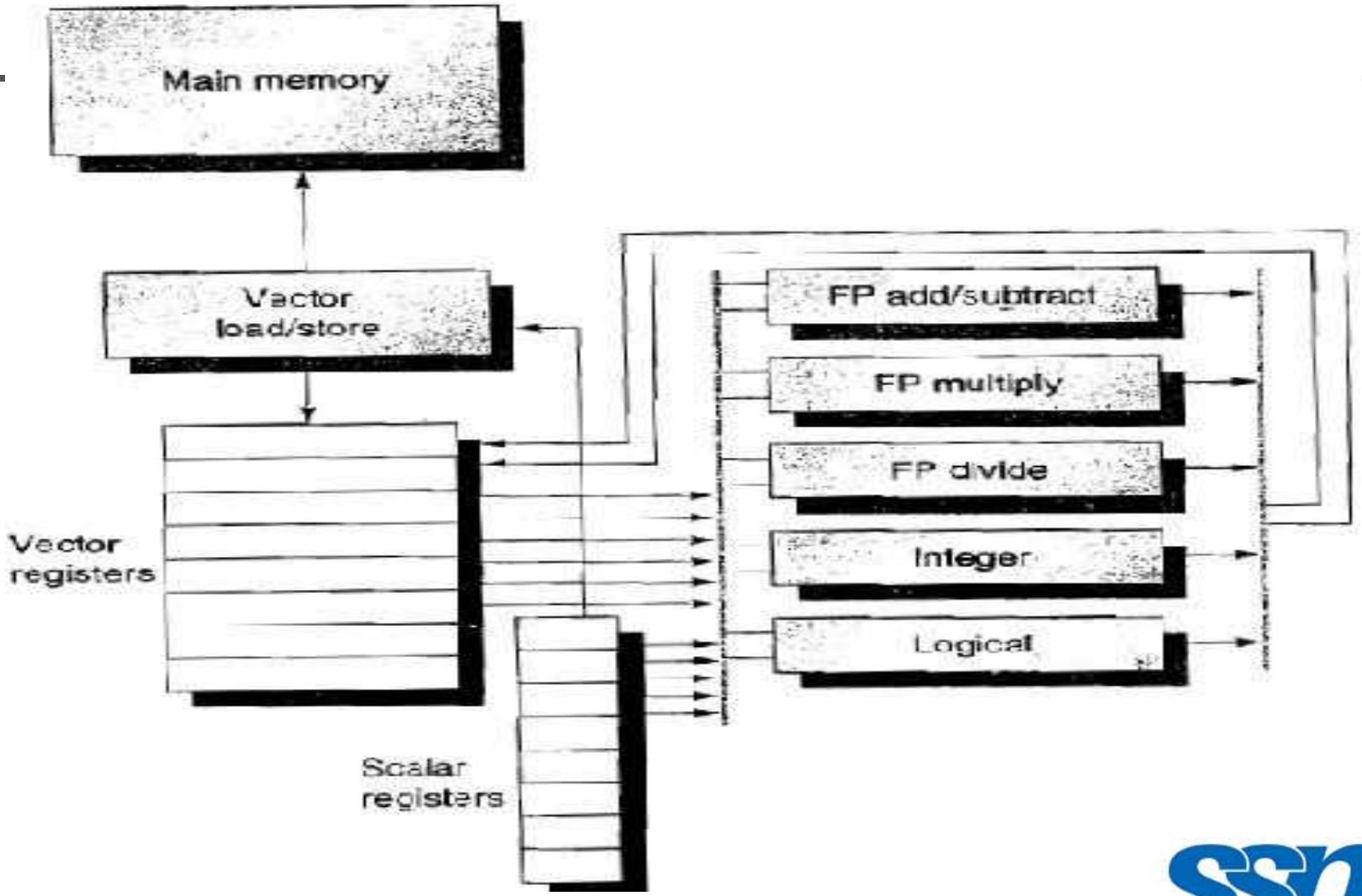
Vector Processor

- 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


Main memory
Vector load/store
Vector registers
Scalar registers
FP add/subtract
FP multiply
FP divide
Integer
Logical

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
- Xand Yare vectors resident in memory
- a is a scalar
- This problem is the so-called SAXPYor DAXPY
  - SAXPY –single precision a* Xplus Y
  - DAXPY-double precision a* Xplus Y

How Vector Processors Work: An Example

- MIPS code for the DAXPY loop

```
    L.D     FO,a        ;load scalar a
    DADDIU  R4,Rx,#512  ;last address to load
Loop: L.D   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

```
L.D     F0,a        ;load scalar a
LV      V1,Rx       ;load vector X
MULVS.D V2,V1,F0    ;vector-scalar multiply
LV      V3,Ry       ;load vector Y
ADDVV.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 n present in the VMIPS code
- compiler produces vector instructions for such a sequence
- resulting code spends its time running in vector mode , the code 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 architects 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.