Implementation of DSP Algorithms

- Main frame computers
- Dedicated (application specific) architectures
- Programmable digital signal processors
  - voice band data modem
  - speech codec

PDSP and General-Purpose MPU

	μPD7720(1980)	8086(1979)	M68000(1980)
Technology	3μNMOS	3μNMOS	3μNMOS
Clock rate	4MHz	8MHz	8MHz
# of Tr.	40k	29k	68k
Bus width	16bit data 23bit inst.	16bit data 16bit inst.	16bit data 16bit inst.
MAC/sec	4M	50k	100k
Power	0.9w	1.7W	1.5w

Recent PDSP

- Eight DSP Core, Each with
  - 1.25 GHz Fixed/Floating-Point Core
    - 40 GMAC/Core for Fixed Point @ 1.25 GHz
    - 20 GFLOP/Core for Floating Point @ 1.25 GHz
  - Memory
    - 32K Byte L1P Per Core
    - 32K Byte L1D Per Core
    - 512K Byte Local L2 Per Core
- Multicore Shared Memory Controller (MSMC)
  - 4096KB MSM SRAM Memory Shared by Eight DSP
- Network Coprocessor
- 0.9 V to 1.2 V Core Supply
- 1.8 V, 1.1 V IO Supply

Features of PDSP

- Support repetitive, numerically intensive tasks
- Powerful data path
- Ability to move large amounts of data to and from memory quickly
- Special instruction set to exploit hardware efficiently

typical example

$$y(n) = h_0 x(n) + h_1 x(n-1) + \cdots + h_{N-1} x(n-N+1)$$

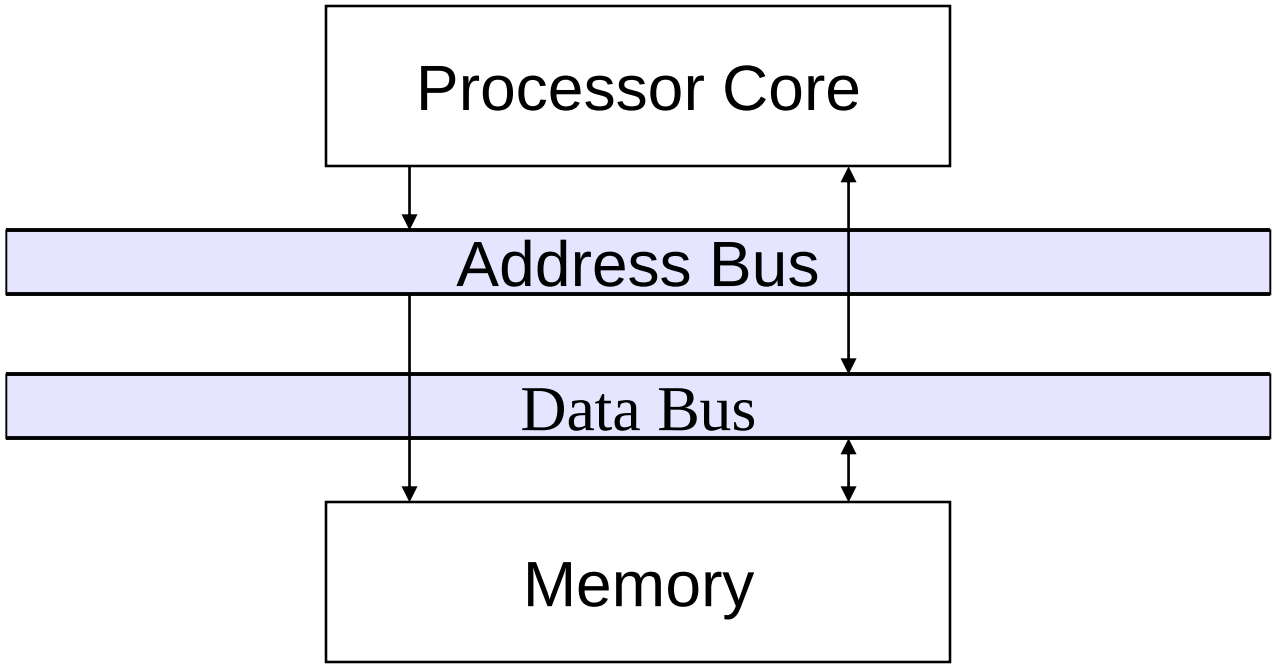
  - fast multiply-accumulation
  - multiple-access memory architecture

Data Path

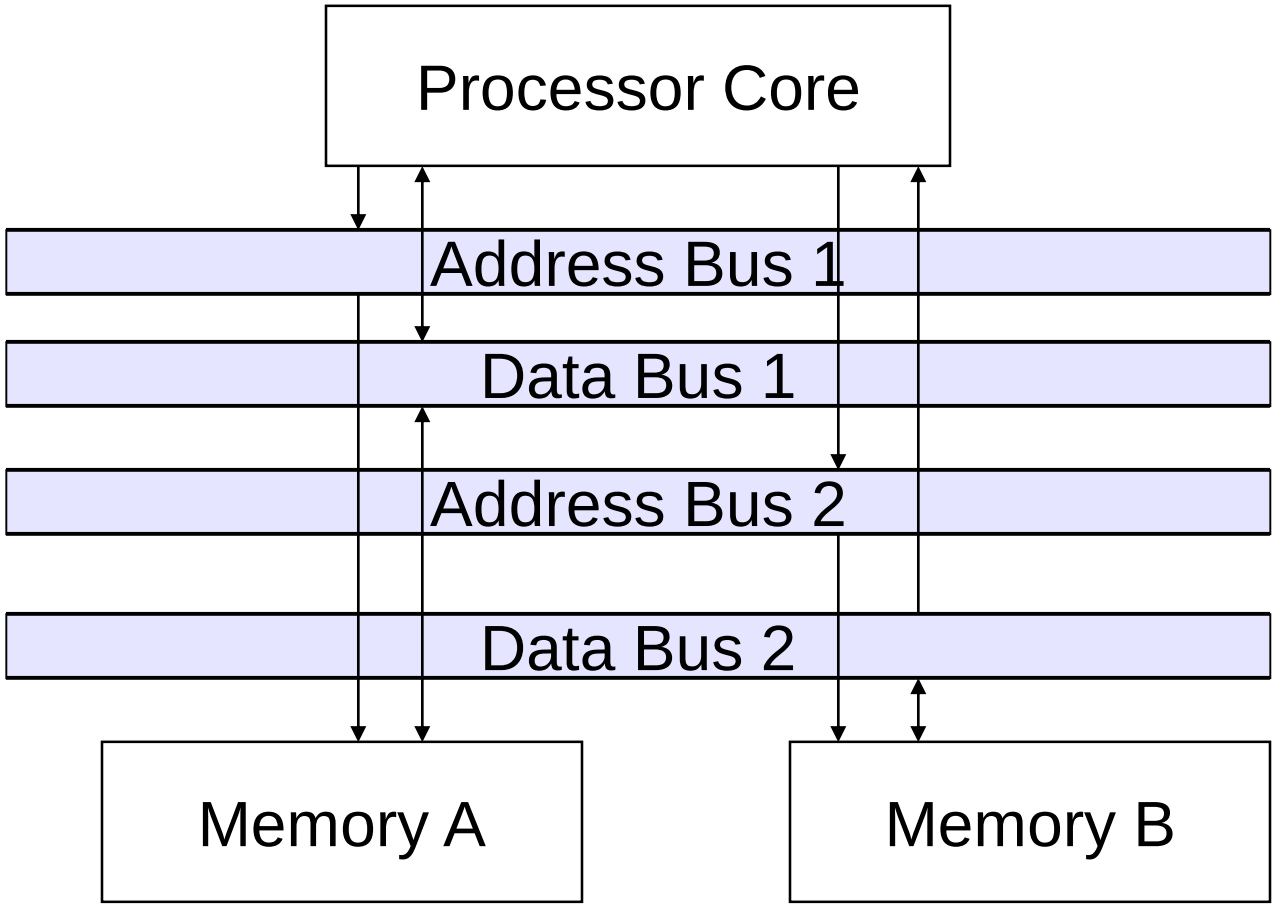
- Multiplier and Accumulator
  - high-speed hardware multiplier
  - integrated with an adder
- ALU
  - basic arithmetic and logic operation
- Shifter (to scale the data by a power of 2)
  - barrel shifter
- Overflow and Saturation
  - overflow can be handled either scaling down the result or by saturation arithmetic

Memory Architecture

- Von Neumann memory architecture

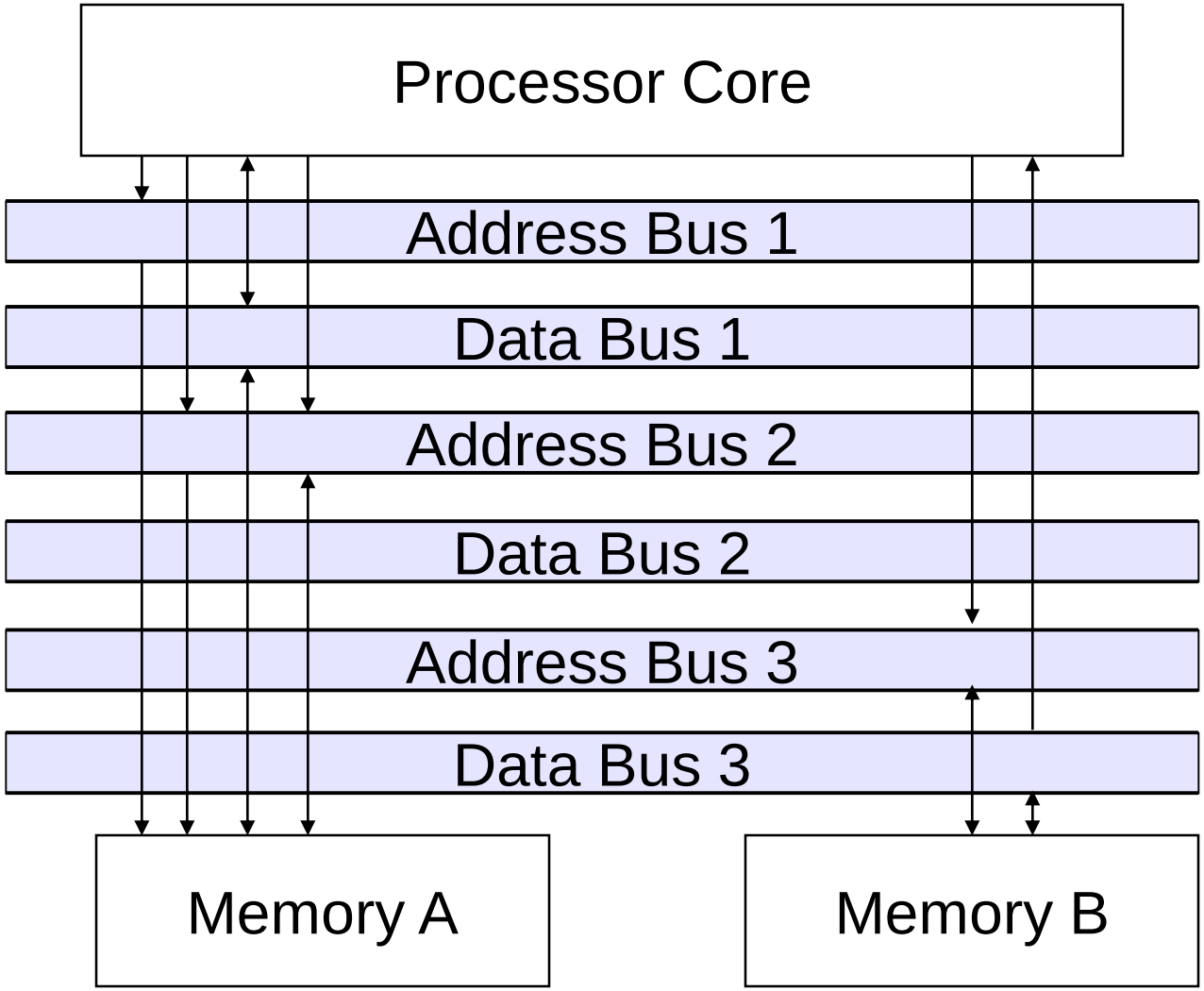


DSP Memory Architecture



Basic Harvard Architecture

Modified Harvard Architecture



dual-ported memory

single-ported memory

Memory Access

Parallel memory banks increase bandwidth

Instructions must specify several memory accesses

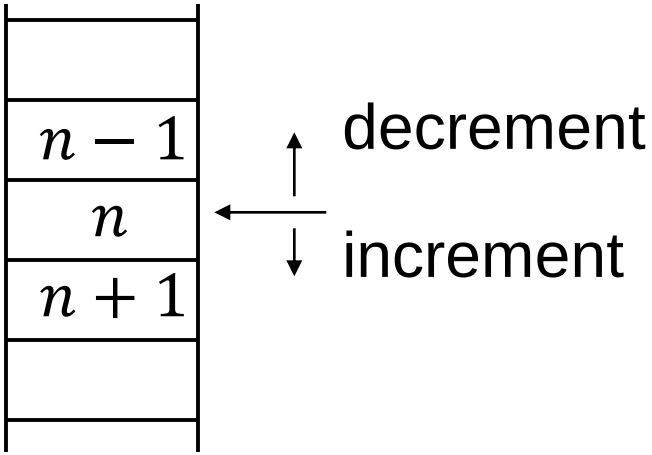
Instruction word gets longer

larger memory space or more memory cycles

simplification needed

Register-Indirect Addressing

- special Address Generation Unit(s)
  - performs one or more complex address calculations per instruction cycle without using the processor's data path
- pre- or post- increment or decrement modes



Pipelining

- A sequence of operations are broken into smaller pieces, which are executed in parallel.
- Pipelining breaks instruction execution into several pipelining stages and allows multiple instructions to be overlapped in execution.
- Pipelining exploits parallelism among the instructions in a sequence of instruction stream, and yields a reduction in the average execution time per instruction.

- Pipelining speeds up the computation,

  but makes programming complicated

Time-Stationary Coding

- Programmer have more explicit control over the pipeline stages.
- An instruction explicitly specifies several operations such as simultaneous operand fetch and execute operations, to be performed in parallel.

```
MAC XO,YO,A   X:(RO)+, XO   Y:(R4)-, YO
a0=a0+p    p=x*y   x=*r0++    y=*r1++
```

Data-Stationary Coding

- A single instruction specifies all the operations performed on a set of operands from memory
- but does not indicate the exact time when these operations are executed.

$$a1 = a1 + (*r5{+}{+} = *r4{+}{+})* *r3{+}{+}$$

- the value in the memory locations pointed by registers r3 and r4 are fetched and multiplied,
- the value that was pointed to by register r4 is written back to the memory location pointed to by register r5, and
- the result of multiplication is accumulated in register a1.

Time-Stationary and Data-Stationary Approaches

Each has advantages and disadvantages

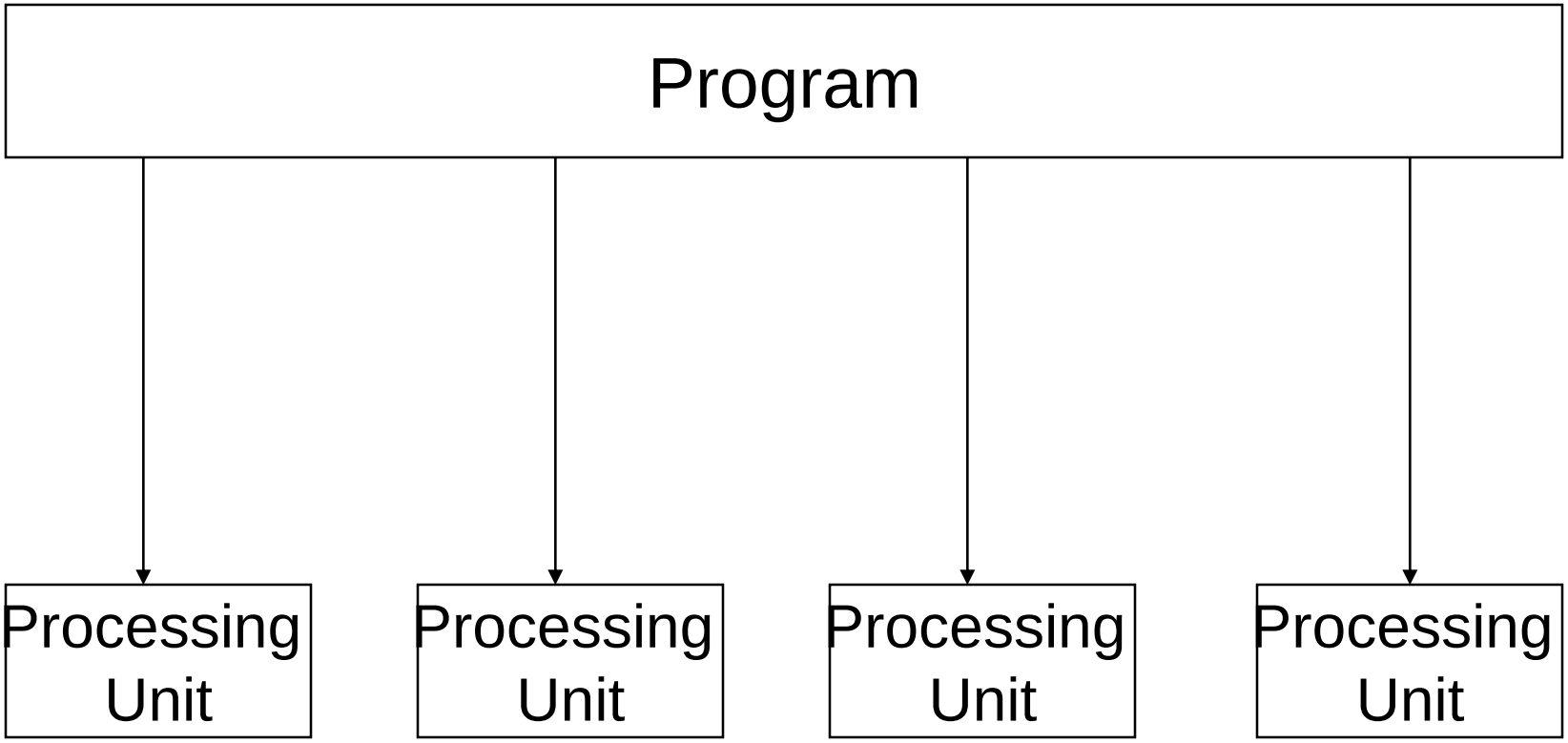
- Time-stationary coding
  - programmers carry out part of the scheduling
- Data-stationary coding
  - programmers specify the operations to be performed, and leave the scheduling tasks to the compiler.
  - easier to read out but not as flexible as time-stationary approach

Processors for Multimedia

audio, speech, image, video, 2D & 3D graphics, etc.

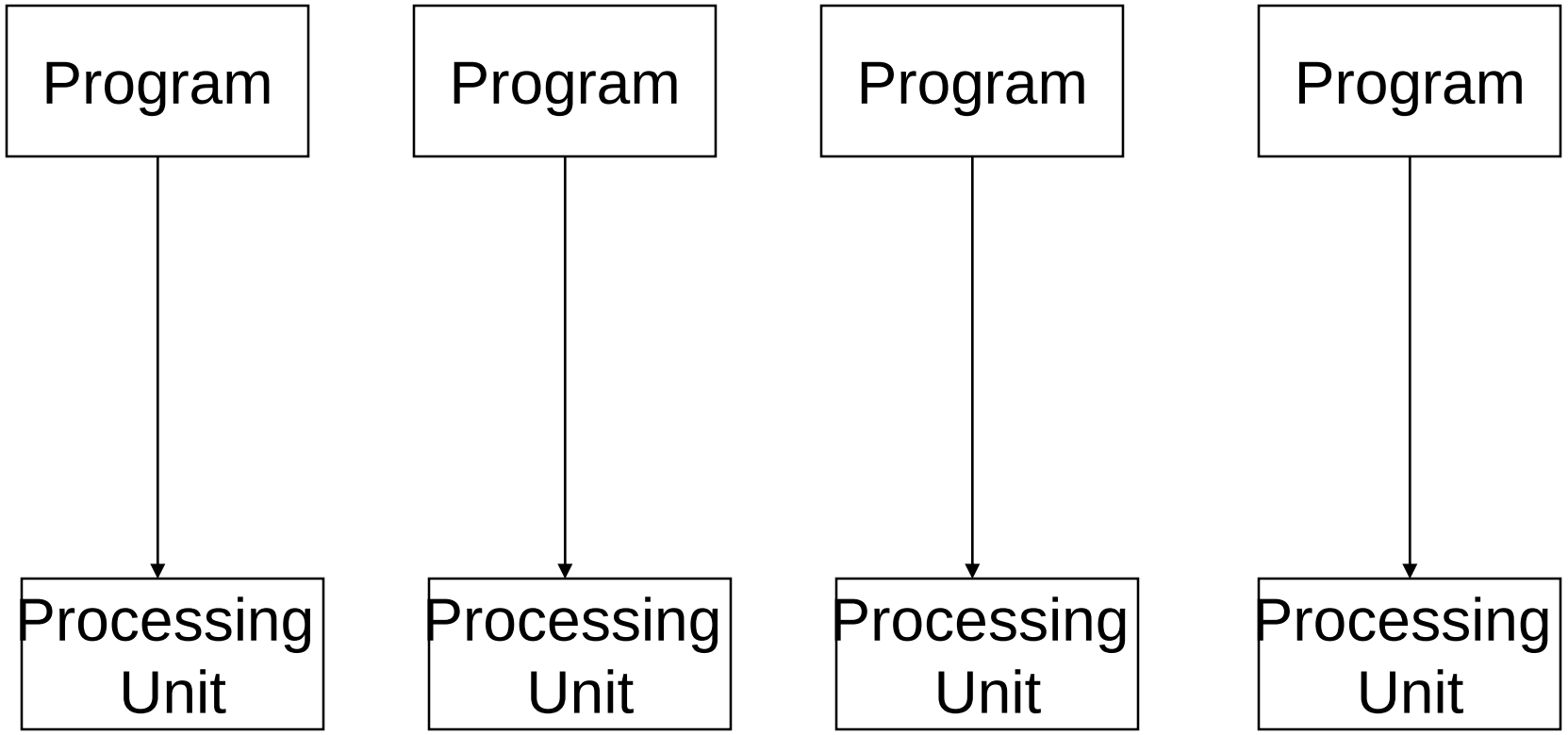
- Frequent use of small integer operands
- Highly regular computation-intensive operations
- Intensive I/O or memory access, data reusability, and data locality
- Complex control operations in less computationally intensive tasks

Multiprocessing



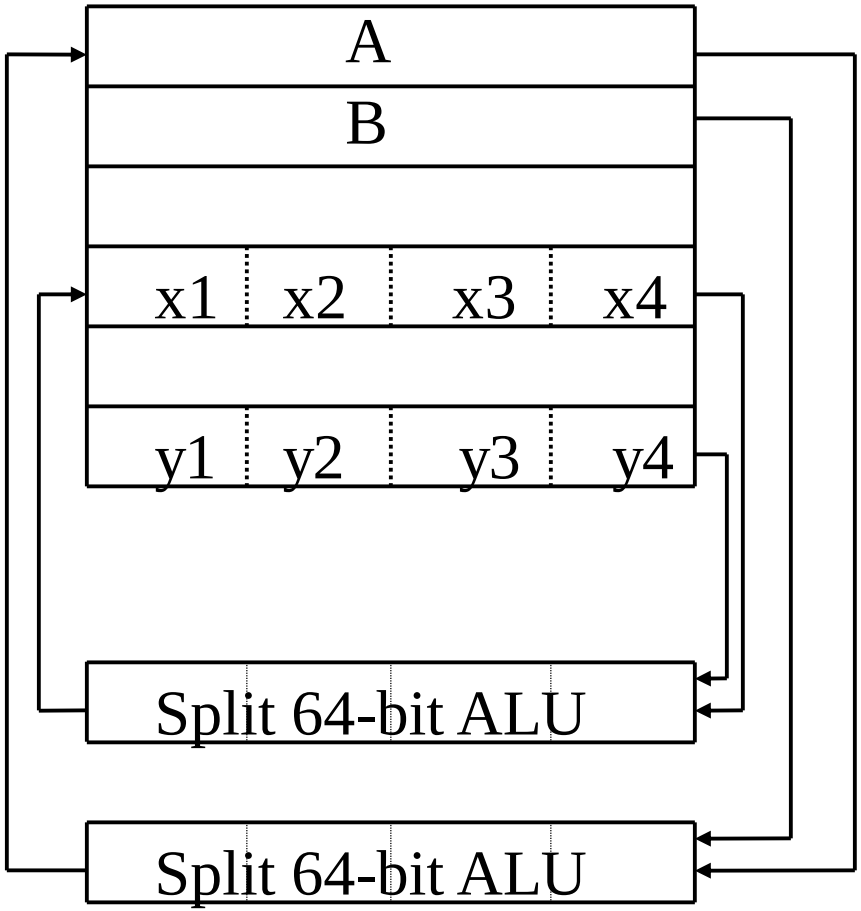
SIMD architecture

Multiprocessing (cont'd)



MIMD architecture

Subword Parallelism



Graphics Processing Unit

- specialized unit designed to accelerate image output in a frame buffer intended for output to a display
- very efficient at manipulating computer graphics and generally more effective than general-purpose CPUs for algorithms where large blocks of data are processed in parallel
- can be used together with a CPU to accelerate general-purpose scientific and engineering applications
- CPU: a few cores optimized for serial processing, GPU: lots of smaller, more efficient cores designed for parallel performance
- TSUBAME 2.5, TSUBAME KFC

Exercise 13

1. Find out the origin of the term “Harvard” architecture.
2. What is VLIW? What are its features in connection with parallelism?
3. Why multicore processors are popular recently?