CS253 Architectures II

Lecture 14

CISC/RISC Vector Processors Parallel Processors

Charles Markham

CS253 Architectures II

Books

"Microprocessors and Interfacing"
D. V. Hall
McGraw-Hill.

"Computer System Architecture"
M. M. Mano
Prentice/Hall International.

"Computer Architecture a Quantitative Approach"
J. Hennessy, D. Patterson
Morgan Kaufmann Publishers.

"The Art of Electronics"
Horowitz and Hill
Cambridge Press.

"Computer Organisation and Architecture"
W. Stallings

Note: There is a large section in the library on computer architecture shelved around 004.22.

Multiply

Many CISC processors include a multiply and divide function in the instruction set, many RISC processors don't. So how do you multiply using a RISC processor?

Decimal

```
  1 2 3 4
        x
     12
  -------
  2 4 6 8
+ 1 2 3 4 0
  -------
  1 4 8 0 8
```

Binary

```
    1 1 0 1
          x
      1 0 1
    -------
    1 1 0 1
+ 0 0 0 0 0
+ 1 1 0 1 0 0
  -----------
  1 0 0 0 0 0 1
```

Multiply

```
        .STARTUP

        mov ax,0
        mov dx,100     ;Multiply dl by bl result in ax.
        mov bx,123
        mov cx,8

back:   rcr dx,1       ;move lsb dx into c
        jnc over
        add ax,bx

over:   shl bx,1       ;multiply bx by 2

        loop back      ;repeat 8 times

                                          mul dl
        call Print
        .EXIT
```

Divide

Decimal

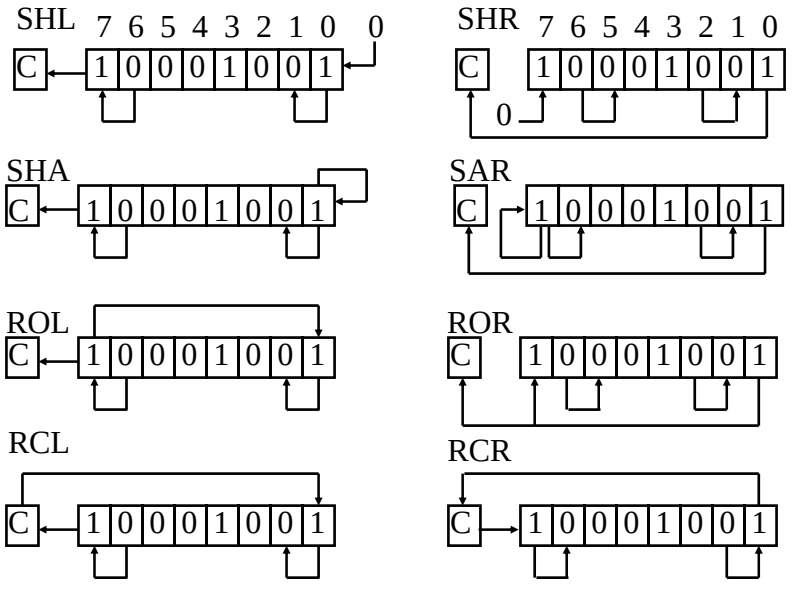
```
        0 1 1 2
    ---------
1 1 ) 1 2 3 4
      0
      ---
      1 2
      1 1
      ---
        1 3
        1 1
        ---
          2 4
          2 2
          ---
            2
```

Binary

```
          0 0 1 0
      -----------
1 0 1 ) 1 1 0 1
        1 0 1
        -----
        0 0 1 1
```

Take the first digit subtract 101, won't go, write down zero, take first and second digit subtract 101, won't go write down zero, take first three digit subtract 101, will go, write down a 1…., repeat until all digits are analysed.

Logical Shift



RISC and CISC

RISC and CISC are two extremes of design philosophy relating to microprocessors.

CISC Complex Instruction Set
Have a large number of operands, addressing modes and possible instructions.
The large number of instructions makes code compiler easier to write but leads to inefficiencies in the decoding of the instruction in the processor.

CISC processors implement micro code inside the processor for maths and data manipulation functions. This makes these specific functions faster than the equivalent RISC processor, however the extra overhead involved in decoding instructions reduces overall performance.
The Pentium/8086 ranges of processors are CISC, each processor is compatible with the one that went before but has added functionality.

RISC

RISC Reduced instruction set microprocessors

These processors normally execute all (or nearly all) there instructions in one clock cycle. To achieve this the designs normally increase processor speed at the expense of available addressing modes and data formats. Any math functions are hardwired rather than implemented with micro code. To get the best out of a RISC system requires high performance compilers (to create efficient code). RISC systems need access to fast memory to provide instructions at a rate that the processor can execute them.

As technology advances, extra complexity on CISC designs is used to create more instructions. RISC designs use the increased complexity to create new registers, pipeline and caches.

RISC

A limited and simple instruction set.

A large number of general purpose registers.

Optimized instruction pipe-line.

Relative Reduction Hardware Cost (Moores Law) has increased software cost, (shortage of programmers). Solution was high level languages (HLL, C++, Java etc). The problem was that there was a big difference between the HLL and the code/hardware. This is referred to as the *semantic gap*. The initial solution was the increasing the functionality of the hardware.

Instruction sets designed to aid the compiler writer. This approach is now known as CISC.

Is there another way of improving performance? Yes RISC.

High Level Languages (Observations)

Operations Performed : Mainly assignment, Conditional, Branch. Sequence control is important.

Operation	Occurrence	Machine Instructions	Memory Access
Assignment	38	13	15
Loop	3	32	26
Call	15	33	45
If	43	21	13
Other	6	1	1

Also 80% of all references were to local variables.

CISC or RISC?

Better Performance

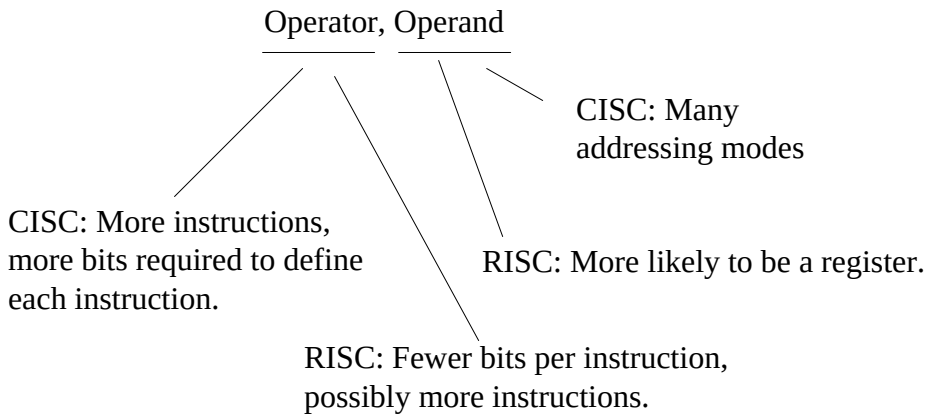
Approach 1 (CISC) make the machine instructions more like the HLL so that fewer are required to execute the program (Compiler Simplification). CISC Smaller faster programs, many compilers use a subset of the instructions.

Approach 2 (RISC) optimise the performance of the most frequently used instructions. Replace memory references with register references. Instruction pipe lines should allow branching to be supported.

RISC makes use of Switch-able register sets (very local cache).

Register optimisation, stores most used variables in registers.

RISC/CISC

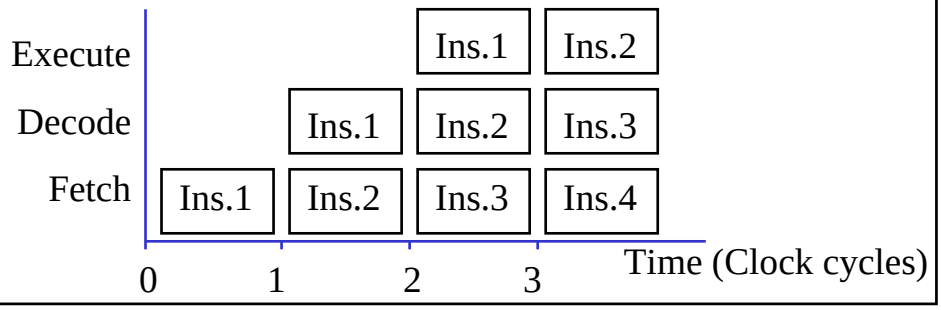


RISC and CISC technologies are converging the differences are not so clear.

Pipelining

All simple processors have to fetch, decode, obtain the operand, execute and return a result to a store for each instruction. Early processors required a separate clock cycle to execute each process and only allowed one instruction to be executed at a time.

Pipeline processors take a conveyer belt approach to executing the instruction. As the first instruction is being executed the second instruction is being decoded and the first being executed.



Pipelining

Branching: The Pentium pipelines fetch both next sequential instruction and speculatively the branched instruction.

Prefetch:	Instructions are stored in a buffer.
Decode 1:	Opcode and Addressing Mode
Decode 2:	Generate addresses for memory access
Execute:	ALU
Write Back:	Result put back into registers, flags updated.

Pentiums contain two execution units U and V. If two instructions are simple they can be executed in Parallel. Simple instructions are hardwired. Super-scalar degree of two.

U and V run in parallel if
Both instructions are simple
No dependencies, result of I1 required by I2

Pipeline Conflicts

Pipelining can cause potential problems

Resource conflicts: Instructions or functions may require use of the same resource such as a bus or memory.

Procedural dependencies: knowing which instruction to execute depends on the outcome of another instruction. Jumps and conditional functions can cause program to jump to instructions not currently in the pipeline.

Data dependencies: one instruction requires the result of another to execute. Thus a subtract may need to know if a carry is occurred when the previous instruction was executed.

Vector Processing

Many problems require repeated calculation on values stored in an array (or vector).

e.g. Add $\begin{bmatrix}1\\2\\3\end{bmatrix}+\begin{bmatrix}3\\4\\5\end{bmatrix}=\begin{bmatrix}4\\6\\8\end{bmatrix}$

$$A+B=C$$

requires 3 separate additions on a scalar machine.

Using parallel ALUs (or processors) as separate units the operation can be executed in parallel.

Vector C = Vector A + Vector B

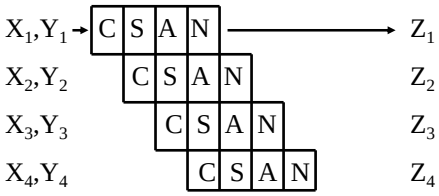
Note: only one control unit.

Vector Processing (Adding Arrays)

C Check if Zero
S Shift significand
A Add
N Normalize

$X = 1.23 \times 10^2$
$Y = 4.56 \times 10^1$
$123.0 + 45.6 = 168.6$
$Z = 1.686 \times 10^2$

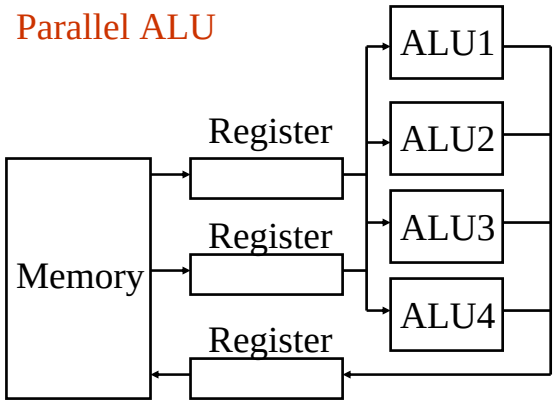
Scalar processing 4*12 = 48 Cycles to produce 12 additions.



Pipeline: 15 Cycles 12 Additions (SAN at the end)

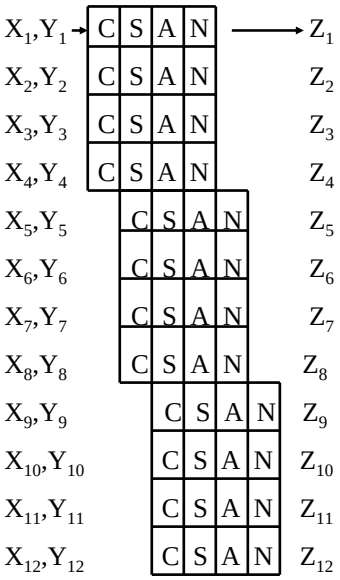
Vector Processing

Parallel ALU



Vector Processors contain registers able to contain the vector.

Cray X-MP



6 Cycles 12 Additions

Vector Processing

Evaluate C = (s*A) + B, where A,B and C are vectors (arrays).

Vector Load	A→VR1
Vector Load	B →VR2
Vector Multiply	s*VR1 →VR3
Vector Add	VR3+VR2 →VR4
Vector Store	VR4 →C

Vector Processing

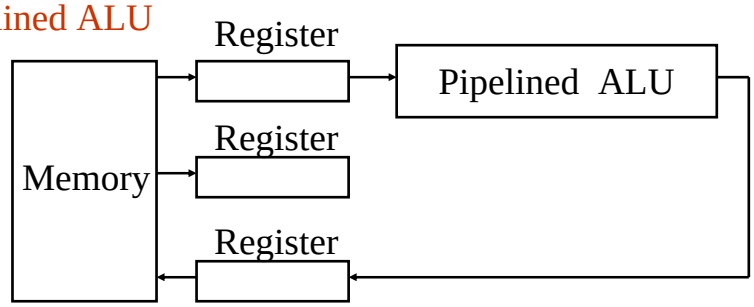
The ALUs in a vector processor can be arranged in series rather than parallel as shown earlier. This creates a pipelined ALU version of the vector processor. In practice this is the more common form of vector processor.

Results can be passed from ALU to ALU in the pipeline. This is known as Chaining. Chaining speed up calculation since intermediate results of a calculation don't have to leave the ALU for further calculation.

Sometimes known as an Array Processor.

Vector Processing

Pipelined ALU



Speedup is caused by

1: The ALU pipeline.

2: The CPU fetches data sequentially for the ALU, note a single instruction will cause the CPU to fetch the entire vector.

An isolated floating point calculation is not speeded up.

Parallel Processing

We have already seen parallel processing at low level in the form of pipelining, multiple ALUs and Co-processors (80387). Although parallel these methods only use one control unit.

Parallel processors contain many control units.

M. Flynn categories for Parallel Processors

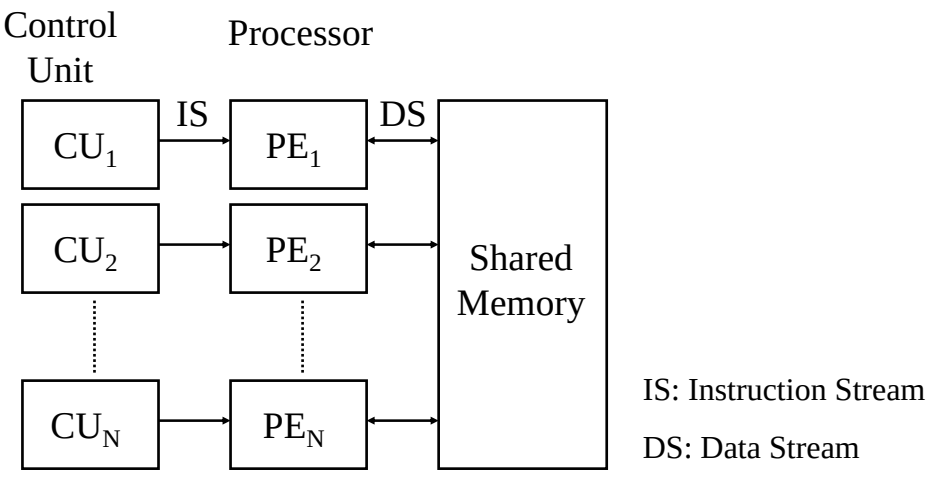
SISD: Single Instruction, Single Data stream. Uses a single processor to interpret single instructions on single pieces of data. Made parallel using pipelining.

SIMD: Single Instruction, Multiple Data stream. Uses a single instruction to control a number of processing elements. Vector processor.

MIMD:Multiple Instruction, Multiple Data stream. A set of processors simultaneously execute different instructions on different sets of data.

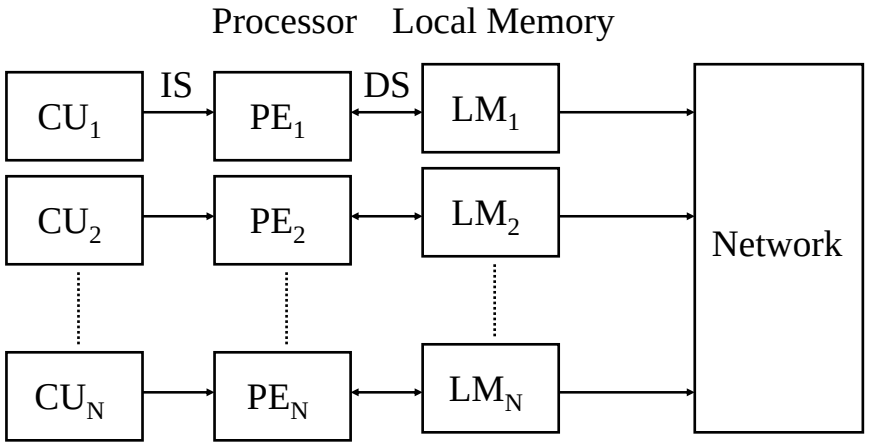
MIMD Processors

Multiprocessor: Shared memory, processors communicate through the shared memory. Very difficult to scale up.



MIMD Processors

Multicomputer: Each processor has its own memory. Dedicated communication path between the processors. Easy to scale up (e.g. Internet).



MIMD Processors

PVM ParallelVirtual Machine

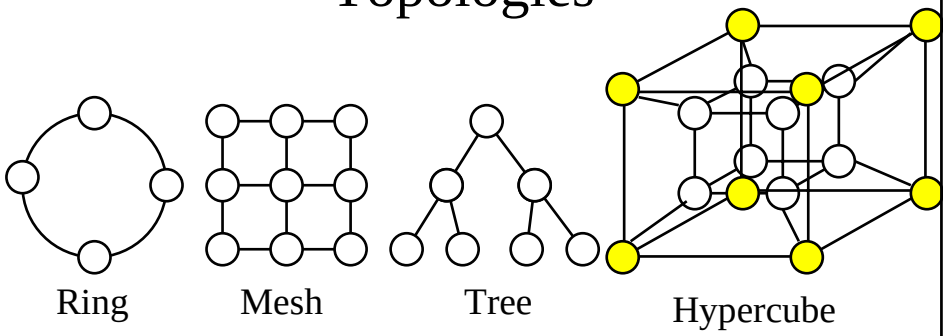
PVM is a message passing system that enables a network of Unix (and NT) computers to be used as a single distributed memory parallel computer. This network is referred to as the virtual machine.

Putting `#include "pvm3.h"` at the beginning of a C program allows distributed computing.

http://www.epm.ornl.gov/pvm/intro.html

Alternatively use Java RMI (Remote Method Invocation)

MIMD Interconnection Topologies



Topologies can be dynamically configurable by the software.

MIMD Interconnection Topologies

Distance to other processors is measured in hops.

Ring: Greatest distance in an n node ring is n/2.

Mesh:(nxn mesh) greatest distance is 2(n-1).

Tree: used for sorting/divide and conquer.

Hypercube: $N=2^n$ processors, each processor has $n=\log_2(N)$ links to adjacent nodes. Diameter is n. Popular since it scales up well.

256 Node Hypercube, Worst case 8 hops, Mesh 32 hops

Coding Examples

Scalar Processing

```
DO 100 I=1,N
DO 100 J=1,N
C(I,J)=0.0
DO 100 K=1,N
C(I,J)=C(I,J)+A(I,K)*B(K,I)
100 CONTINUE
```

Vector Processing

```
DO 100 I=1,N
C(I,J)=0.0 (J=1,N)
DO 100 K=1,N
C(I,J)=C(I,J)+A(I,K)*B(K,I)(J=1,N)
100 CONTINUE
```

Note: (J=1,N) means do all at once, as a vector calculation.

Coding Examples

Parallel Processing

```
    DO 50 J=1,N
    FORK 100
50  CONTINUE
    J=N
100 DO I=1,N
    C(I,J)=0.0
    DO 200 K=1,N
    C(I,J)=C(I,J)+A(I,K)*B(K,I)(J=1,N)
200 CONTINUE
    JOIN N
```

The FORK spawns separate processes for the code that follows (until the next FORK).

JOIN waits for N processes to complete before continuing.

Superscalar Processors

It is possible to create a processor that will execute two or more instructions simultaneously. This is known as instruction-level parallelism. Compilers need to be aware of very many possible conflicts.

Data dependancy: add r1,r2 move r3,r1

Procedural Dependancy: Instructions following a conditional branch can't be executed until the branch instruction is executed.

Resource Conflicts: mov r1,r3 mov r1,r2 both require r1

Personal-Computer Systems (PC)

PC: Computer System dedicated to a single user.

I/O: Keyboard, Screen, Printer, Mouse, Speakers, Microphone, Modem etc.

PC Systems still require memory management, file protection and multitasking. The design ideas for single user systems have been transfered from the earlier OS designs for large many-user systems.

e.g. MULTICS (GE645 Mainframe)-> UNIX (Workstation) -> Windows NT (PC)

Note: The OS is not designed to maximise CPU usage but to maximise the convenience and responsiveness of the computer to the user.

Parallel Systems

Most computers contain only one processor. The trend is towards multiprocessor systems, computers with more than one processor.

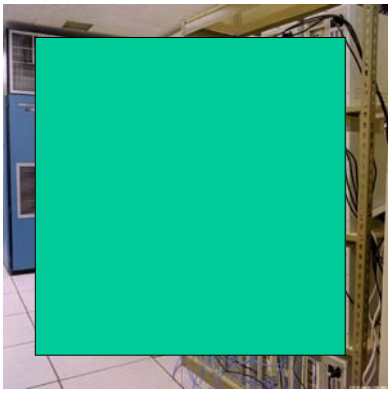
Tightly Coupled:	Processors share buses, clocks and peripherals. Communication is via shared memory. Difficult system to scale for large numbers of processors. (See SE203 for more detail)
Fault Tolerance:	If a single processor on a parallel system fails then in theory at least the other process can keep programs running (all be it a little slower). This is known as graceful degradation.

Parallel Systems

Tandem Systems:	Two copies of each process are run simultaneously on two processors. At fixed times the primary process (and data is copied to the backup machine). If the primary system fails then the backup is used to restart it from the last backup. Costly!
Symmetric Systems:	Each processor runs an identical copy of the OS. The various processes communicate as needed. UNIX can support symmetric systems.
Asymmetric Systems:	Each processor is assigned a specific task. A master processor controls the overall allocation of tasks to the slave processors.

Parallel Systems

In a Distributed System the Processors the processors do not share common memory or clock. Communication between processes is via dedicated communication lines. A distributed system is also known as loosely coupled or a multicomputer system.



Beowulf,

LOKI 16 computer cluster, with a total of 3200 MHz and 2Gb of RAM. This machine is about 1/10th the cost of an equivalent supercomputer. Each computer in it would cost you less than $1000 today.

Benefits of Parallel Systems

Resource Sharing:	*Users at one site can use resources at another site. (e.g. Printer)*
Computation Speedup:	*Concurrent run of different parts of the same program allows for speed up. Heavily loaded sites can load_share with free machines.*
Reliability:	*If there is sufficient redundancy in the system then the failure of one site will not stop the system.*
Communication:	*Allows information to shared over large distances. (e.g. Windows Network Neighbourhood, E-mail etc.)*