Processor and Computer Architecture

Differences between RISC and CISC processors

Importance/use of pipelining and registers in RISC processors

Interrupt handling on CISC and RISC processors

Computer architectures: SISD,SIMD,MISD,MIMD

Characteristics of massively parallel computers

The control unit

- While a program is being executed, the CPU is receiving a sequence of machine-code instructions
- It is the responsibility of the control unit within the CPU to ensure that each machine instruction is handled correctly
- There are two ways that a control unit can be designed to allow it to perform its function

The control unit

- Construct as a logic circuit (this is the hardwired solution where the machine-code instructions are handled directly by hardware)

The control unit

- Use **microprogramming**
- i.e. The control unit contains a ROM component in which is stored the microinstructions or microcode for microprogramming
- This is often referred to as **firmware**
- The choice of which method is used is largely dependent on the type of processor

CISC and RISC processors

- Architecture involves the following:
  - the instruction set
  - the instruction format
  - the addressing modes
  - the registers accessible by instructions
- Choice of the instruction set is the main factor in deciding on a suitable architecture

CISC and RISC processors

- One school of thought believes that the instruction set should be chosen so that:
  - It can be clearly applied to important problems,
  - That only simple equipment is required, and
  - That important problems are handled speedily
- A second school of thought emphasizes that it should be chosen to suit the needs of high level languages

CISC and RISC processors

- Architecture of first computer systems concentrated on the needs of high level languages
- Contained what CISC (Complex Instruction Set Computers) processors
- Complexity increased with the evolution of computer systems
- E.g. PDP-11; VAX; Motorola 6800, 6809 and 68000-families; the Intel 8080 and x86-family; the Zilog Z80, Z8 and Z8000-families and many others
- However, this was highly challenged in the late 70's

CISC and RISC processors

- Engineers and other scientists believed that a "smaller" instruction set could be more suitable and efficient
- Introduced RISC (Reduced Instruction Set Computers)
- This led to the development of RISC processors
- E.g. DEC Alpha, AMD Am29000, ARC, ARM, Atmel AVR, Blackfin, Intel i860 and i960, MIPS, Motorola 88000, PA-RISC, Power (including PowerPC), RISC-V, SuperH, and SPARC

CISC and RISC processors

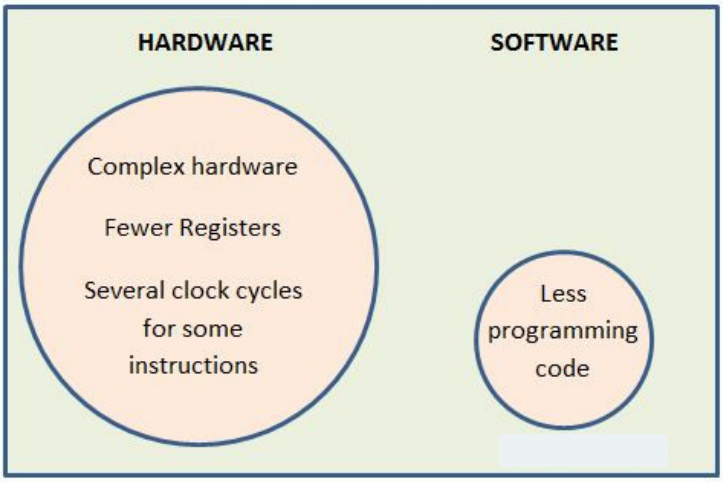
- Many smartphones and tablets, such as iPad and a great number of Android based devices, use ARM processors
- Supercomputers also use RISC processors such as the DEC Alpha and Sequoia processors

CISC and RISC processors

- The simplicity of the instructions allows data to be:
  - Stored in registers and
  - Manipulated in them with no resource to memory access other than that necessary for initial loading and possible final storing
- The simplicity also allows hard-wiring inside the control unit with limited complexity required

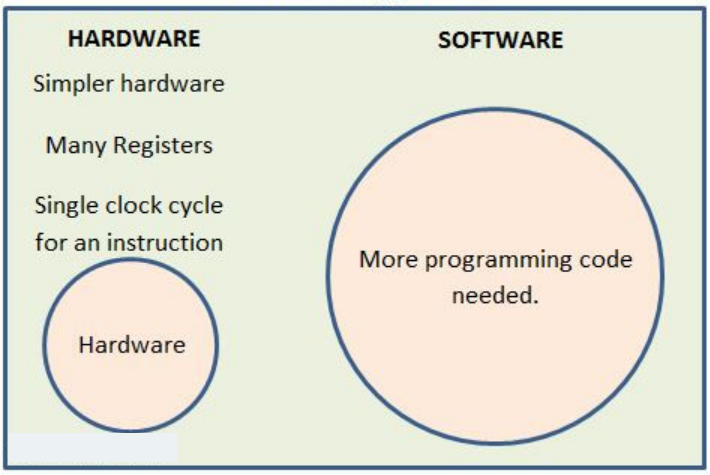
CISC and RISC processors

The CISC approach



CISC and RISC processors

The RISC approach



CISC and RISC processors

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers

CISC and RISC processors

RISC	CISC
Fewer instructions	More instructions
Simpler instructions	More complex instructions
Small number of instruction formats	Many instruction formats
Single-cycle instructions whenever possible	Multi-cycle instructions
Fixed-length instructions	Variable-length instructions
Only load and store instructions to address memory	Many types of instructions to address memory
Fewer addressing modes	More addressing modes

CISC and RISC processors

- RISC processors provide efficient pipelining
- Pipelining is a form of parallelism applied specifically to instruction execution
- **Pipelining** is that the fetch-decode-execute cycle can be separated into a number of stages
  - instruction fetch (IF)
  - instruction decode (ID)
  - operand fetch (OF)
  - instruction execute (IE)
  - result write back (WB)

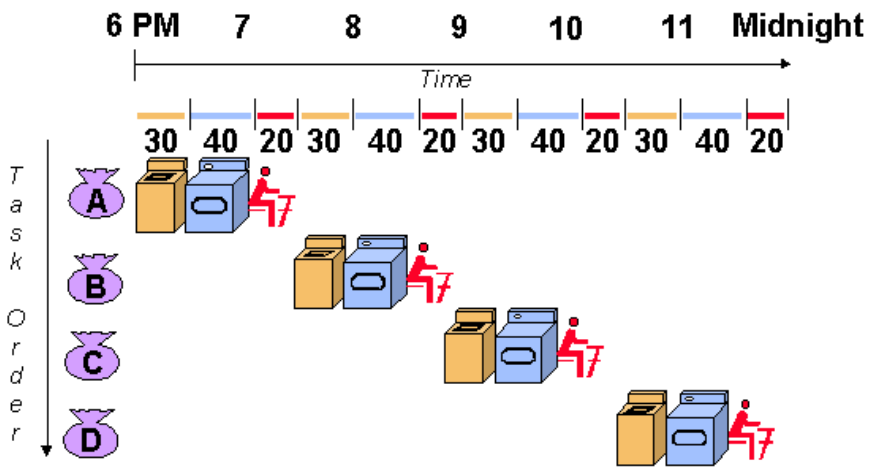
CISC and RISC processors

Pipelining

- Using the standard fetch-decode-execute cycle, an instruction can be:
  - fetched (from memory),
  - decoded (by the control unit) or
  - executed (by the control unit)
- One approach would be to split the processor up into three parts, each of which handles one of the three stages

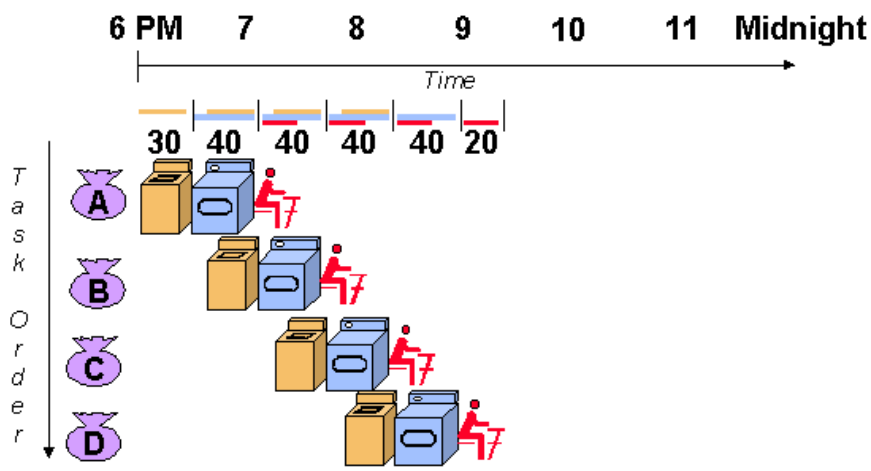
CISC and RISC processors

Pipelining



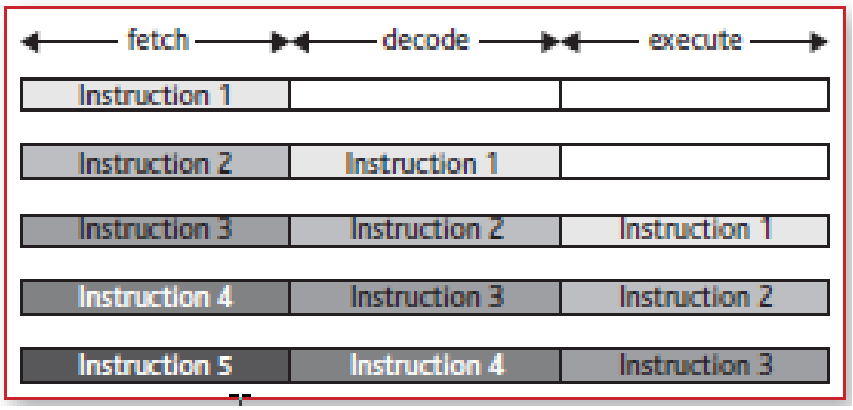
CISC and RISC processors

Pipelining



CISC and RISC processors

Pipelining



CISC and RISC processors

Pipelining

- Pipelining increases the speed of throughput of program instructions, unless the next instruction in the pipe is not the next one that is needed
- Suppose Instruction 2 is a jump to Instruction 10
- Then Instructions 3, 4 and 5 need to be removed from the pipe and Instruction 10 needs to be loaded into the fetch part of the pipe
- Thus, the pipe has to be cleared and the cycle restarted

CISC and RISC processors

Pipelining

← fetch →	← decode →	← execute →
Instruction 1		
Instruction 2	Instruction 1	
Instruction 3	Instruction 2	Instruction 1
Instruction 4	Instruction 3	Instruction 2
Instruction 10		
Instruction 11	Instruction 10	
Instruction 12	Instruction 11	Instruction 10

CISC and RISC processors

- Pipelining
  - instruction fetch (IF)
  - instruction decode (ID)
  - operand fetch (OF)
  - instruction execute (IE)
  - result write back (WB)

Processor units	Clock cycles 1	2	3	4	5	6	7
IF	1.1	2.1	3.1	4.1	5.1	6.1	7.1
ID		1.2	2.2	3.2	4.2	5.2	6.2
OF			1.3	2.3	3.3	4.3	5.3
IE				1.4	2.4	3.4	4.4
WB					1.5	2.5	3.5

Interrupts

- An interrupt is a signal sent to the processor from a hardware device, indicating that the device requires attention
- One is sent, for example, when a key has been pressed or when one of the software timers needs updating
- This sending of a signal is known as an interrupt request
- RISC OS deals with the interrupt by temporarily halting its current task, and entering an interrupt routine

Interrupts

- This routine deals with the interrupting device very quickly - so quickly, in fact, that you will never realise that your program has been interrupted
- Interrupts provide a very efficient means of control since the processor doesn't have to be responsible for regularly checking to see if any hardware devices need attention
- Instead, it can concentrate on executing your code or whatever else its current main task may be, and only deal with hardware devices when necessary

Interrupts

- Amongst the devices which are handled under interrupts on RISC OS computers are the:
  - Keyboard
  - Printer
  - USB port
  - Mouse
  - Disc drives
  - Built-in timers
- Additionally, external hardware such as expansion cards may cause new interrupts to be generated

Interrupts

- Each potential source of interrupts has a device number
- There are corresponding device vectors; installed on each vector there is a default device driver that receives only the interrupts from that device
- Unless you are adding your own interrupt-generating devices to the computer, you should not need to alter the interrupt system

Parallel processing

Array processor

- Involves more than one arithmetic and logic unit but still only one processor
- This is particularly useful when processing data held in a one dimensional array when the same operation is to be applied to every element of the array
- An example could be where all the values represent the cost of different stock items and it is required to compute selling prices calculated by adding the same percentage to each price value
- An array processor is able to do the same calculation simultaneously to each of the input values

Parallel processing

Parallel processor

- An extension of array processors
- This system uses many independent processors working in parallel on the same program
- One of the difficulties with this is that the programs running on these systems need to have been written specially for a parallel processor architecture
- If the programs have been written for standard processor architectures, some instructions cannot be processed until others have been completed
- Thus, checks have to be made to ensure that all prerequisites have been completed

Parallel processing

Parallel processor

- These systems are in use, particularly in systems that receive many inputs from sensors and the data needs to be processed in parallel
- A simple example that shows how the use of parallel processors can speed up a solution is the summing of a series of numbers
- Consider finding the sum of n numbers. A single processor would involve (n − 1) additions in sequence

Parallel processing

Parallel processor

- Using n/2 processors, we could simultaneously add n/2 pairs of numbers in the same time it would take a single processor to add one pair of numbers
- This would leave only n/2 numbers to be added and this could be done using n/4 processors
- Continuing in this way, the time to add the series would be considerably reduced

Parallel processing

- One computer can have multiple processors running in parallel
- In principle, there are four categories of system:
  - SISD (Single Instruction Single Data stream)
  - SIMD (Single Instruction Multiple Data stream)
  - MISD (Multiple Instruction Single Data stream)
  - MIMD (Multiple Instruction Multiple Data stream)

Parallel processing

- **SISD (Single Instruction Single Data stream)** is the typical arrangement found in early personal computers
- There is a single processor so no processor parallelism
- The single data stream just means one memory

- **SIMD (Single Instruction Multiple Data stream)** describes how an array or vector processor works
- The multiple processors each have their own memory
- One instruction is input and each processor executes this instruction using data available in its dedicated memory

Parallel processing

- **MISD (Multiple Instruction Single Data stream)** isn't implemented in commercial products
- **MIMD (Multiple Instruction Multiple Data stream)** has examples in modern personal computers which are of the symmetric multiprocessor type using identical processors
- In this case, each processor executes a different individual instruction
- The multiple data stream can be provided by a single memory suitably partitioned
- Each processor might have a dedicated cache memory

Parallel processing

Parallel computer systems

- Examples of one type of multicomputer system are called massively parallel computers
- These are the systems used by large organisations for computations involving highly complex mathematical processing
- They are the latest in an evolution of what have traditionally been called 'supercomputers‘

Parallel processing

- The major difference in architecture is that instead of having a bus structure to support multiple processors there is a network infrastructure to support multiple computer units
- The programs running on the different computers can communicate by passing messages using the network
- An alternative type of multicomputer system is **cluster computing**, where a very large number of PCs are networked