# K. S. Institute of Technology

## Department of Computer Science and Engineering

**Advanced Computer Architecture – 18CS733**

**Module-2**

**Faculty Name: Dr. Vijayalaxmi Mekali**

**Associate Professor, Dept. of CSE**

**KSIT, Bangalore**

# Module 2
# Hardware Technologies

## Processors and Memory Hierarchy

### Design Space of Processors

- Processors can be mapped to a space that has clock rate and cycles per instruction (CPI) as coordinates. Each processor type occupies a region of this space.
- As implementing technology evolve rapidly, the clock rate of various processor have moved from lower to higher speed to towards the right of the design space.
- Newer technologies are enabling higher clock rates.
- Manufacturers are also trying to lower the number of cycles per instruction (CPI).
- Thus the future processor space is moving toward the lower right of the processor design space.

# Module 2
# Hardware Technologies

## Design Space of Processors cond…

- **Processor manufactures have been trying to lower the CPI rate using innovative hardware approaches.**

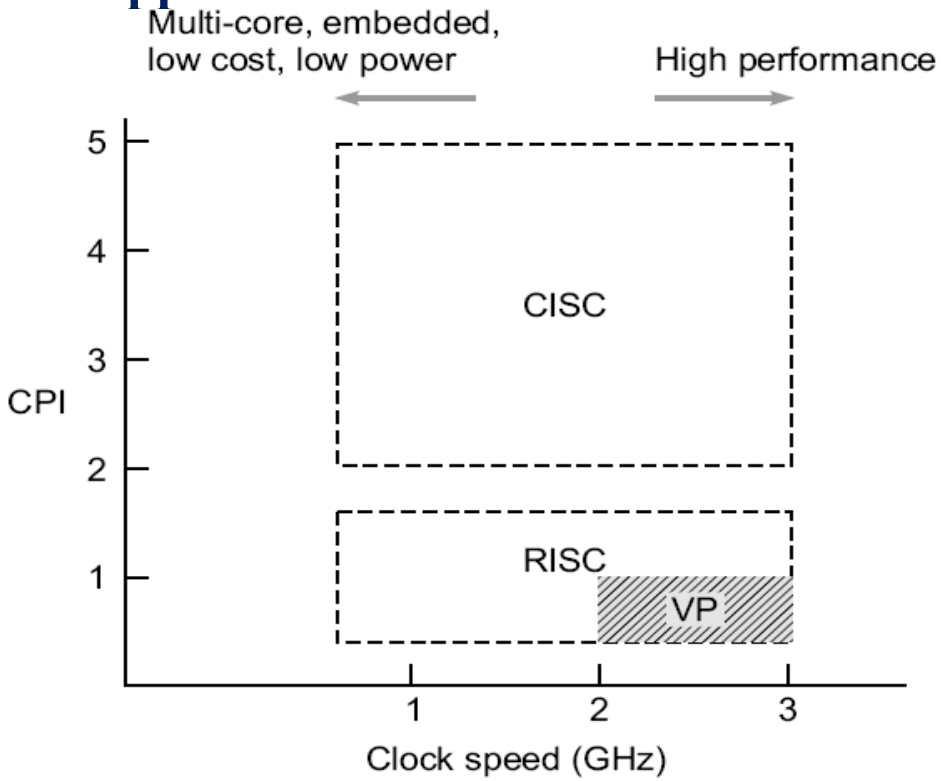



**Fig. 4.1** CPI versus processor clock speed of major categories of processors

# Module 2
# Hardware Technologies

## Design Space of Processors cond…

- The fig shows the broad CPI vs clock speed characteristics of major categories of the current processor. (CISC and RISC).
- CISC (Complex Instruction Set Architecture) and RICS (Reduced Instruction Set Architecture) are designed for multi core chips, embedded applications or for low cost and low power consumption.
- RISC processor example MIPS, SPARC etc.

**Design space- CISC and RISC Processors**

**CISC**

- CISC processor examples are Intel Pentium, IBM 360, x86. Theses processor have advanced technology with clock speed up to few GHz. CPI of different instructions varies from 1 to 20 cycles.

# Module 2

## Hardware Technologies

Hard wired approach is designed for RISC systems. Execution of instruction is controlled by control signals

### Design space- CISC and RISC Processors

### RISC

- **Reduced Instruction Set Computing (RISC) processors like the Intel i860, SPARC, MIPS R3000, and IBM RS/6000, which have hard-wired control units, higher clock rates, and lower CPI figures.**
- **These processor have efficient pipelines, average CPI in between one and two cycles**

### 1. Supperscalar processors

  - **Important class of RISC processor are Superscalar Processors which allow the multiple instructions to be issued simultaneously during each cycle.**
  - **Thus the CPI of supperscalar processor is less than scalar RISC processors.**
-

# Module 2
# Hardware Technologies

## Design space- CISC and RISC Processors

### 2. VLIW Machines

- Very Long Instruction Word machines typically have many more functional units than superscalars (and thus the need for longer – 256 to 1024 bits – instructions to provide control for them).
- These machines mostly use microprogrammed control units with relatively slow clock rates because of the need to use ROM to hold the microcode.

### 3. Superpipelined Processors

These processors typically use a multiphase clock (actually several clocks that are out of phase with each other, each phase perhaps controlling the issue of another instruction) running at a relatively high rate.

- The CPI in these machines tends to be relatively high (unless multiple instruction issue is used).
- Processors in vector supercomputers are mostly superpipelined and use multiple functional units for concurrent scalar and vector operations.

## Instruction Pipeline

**Instruction execution includes four plases:**

- **fetch**
- **decode**
- **execute**
- **write-back**

– **These four phases are frequently performed in a pipeline, or —assembly line manner, as illustrated on the figure 4.2.**

– **The pipeline, like an industrial assembly line, receives successive instructions from its input end and executes them in a streamlined, overlapped fashion as they flow through.**

– **A pipeline cycle is intuitively defined as the time required for each phase to complete its operation, assuming equal delay in all phases (pipeline stages).**

# Instruction Pipeline cond…

## Base Scalar processor

- **A pipeline cycle is defined as time required for each phase to complete its operation assuming equal delay in all phases.**

1. **Instruction pipeline cycle: the clock period of the instruction pipeline.**
2. **Instruction issue latency: The time (in cycles) required between the issuing of the two adjacent instructions.**
3. **Instruction issue rate: The number of instructions issued per cycle and also called degree of a superscalar processor.**
4. **Simple operation latency: Simple operation makes up the vast majority of instruction executed by the machine, such as adds, loads, stores, branches etc.**

- **Resource conflict: Two or more instructions demands the same resources on same clock cycle.**

# Module 2
# Hardware Technologies

- C]

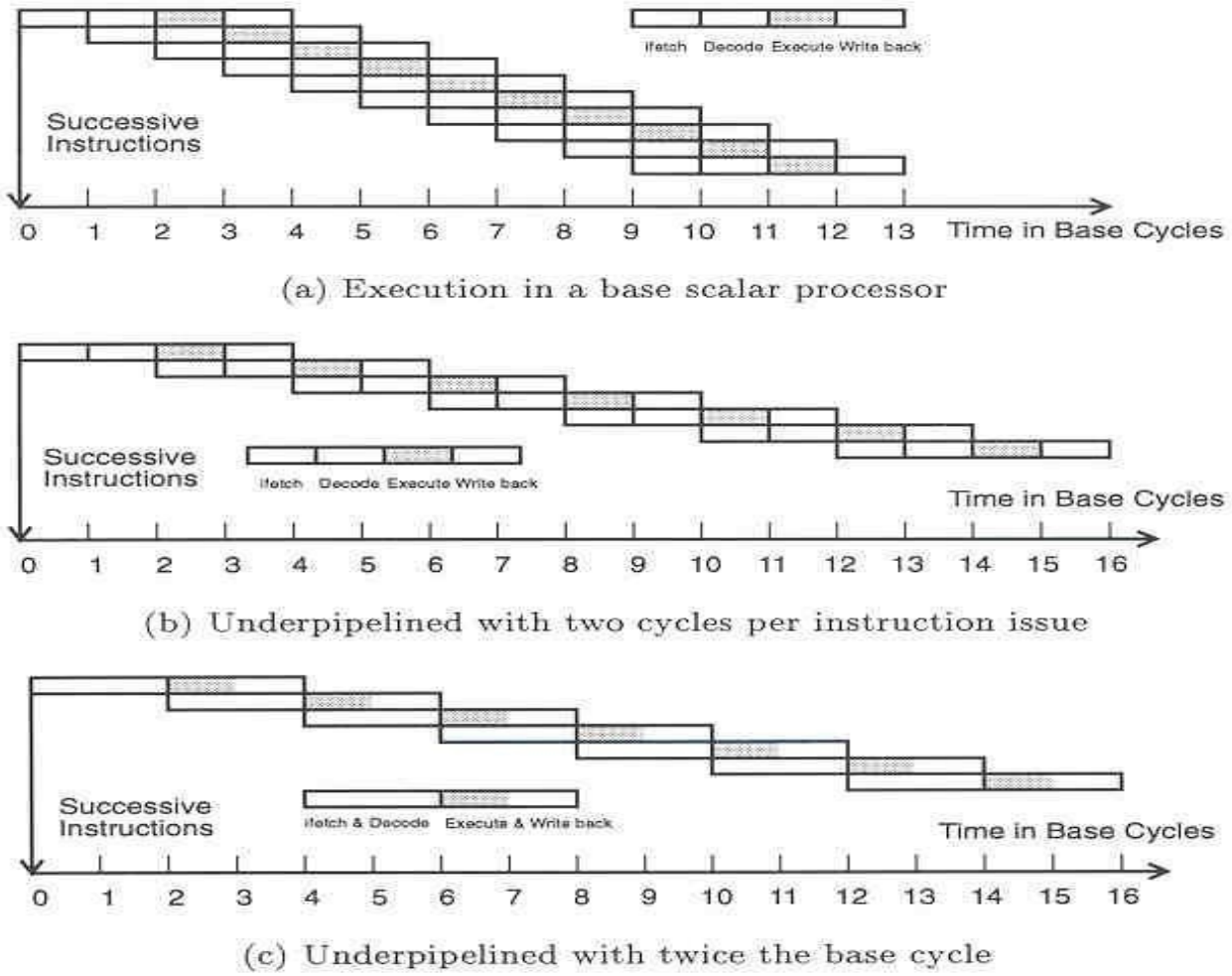



**Figure 4.2 Pipelined execution of successive instructions in a base scalar processor and in two underpipelined cases.** Courtesy of Jouppi and Wall; reprinted from *Proc. ASPLOS*, ACM Press, 1989)

## Instruction Pipeline cond…

- In fig 4.2a only one instruction is issued per cycle, ie one cycle latency between the instructions.
- Fig 4.2b shows issue latency of two cycles.
- Fig 4.2c shows pipeline cycle is doubled by combining pipeline stages fetch and decode are combined, execute and writeback are combined.

## Instruction Pipeline cond…

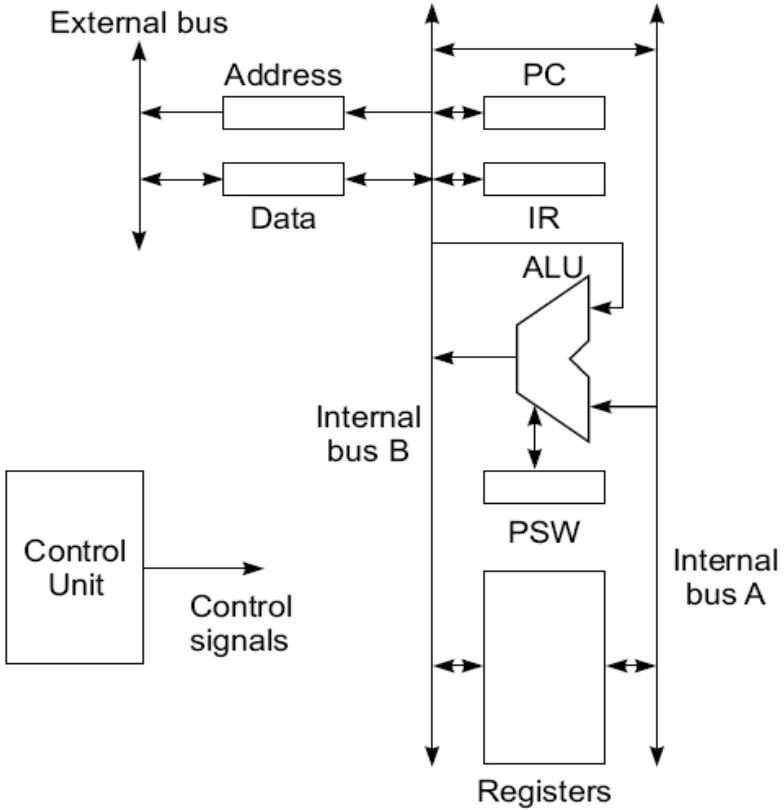



**Fig. 4.3** Data path architecture and control unit of a scalar processor

- Figure 4.3 shows the data path architecture and control unit of a typical, simple scalar processor which does not employ an instruction pipeline. Main memory, I/O controllers, etc. are connected to the external bus.
- The control unit generates control signals required for the fetch, decode, ALU operation, memory access, and write result phases of instruction execution.

- **Instruction Pipeline cond…**
- **The control unit itself may employ hardwired logic, or as was more common in older CISC style processors microcoded logic.**
- **Modern RISC processors employ hardwired logic, and even modern CISC processors make use of many of the techniques originally developed for high-performance RISC processors.**

# Module 2

## Hardware Technologies-Instruction Set Architectures

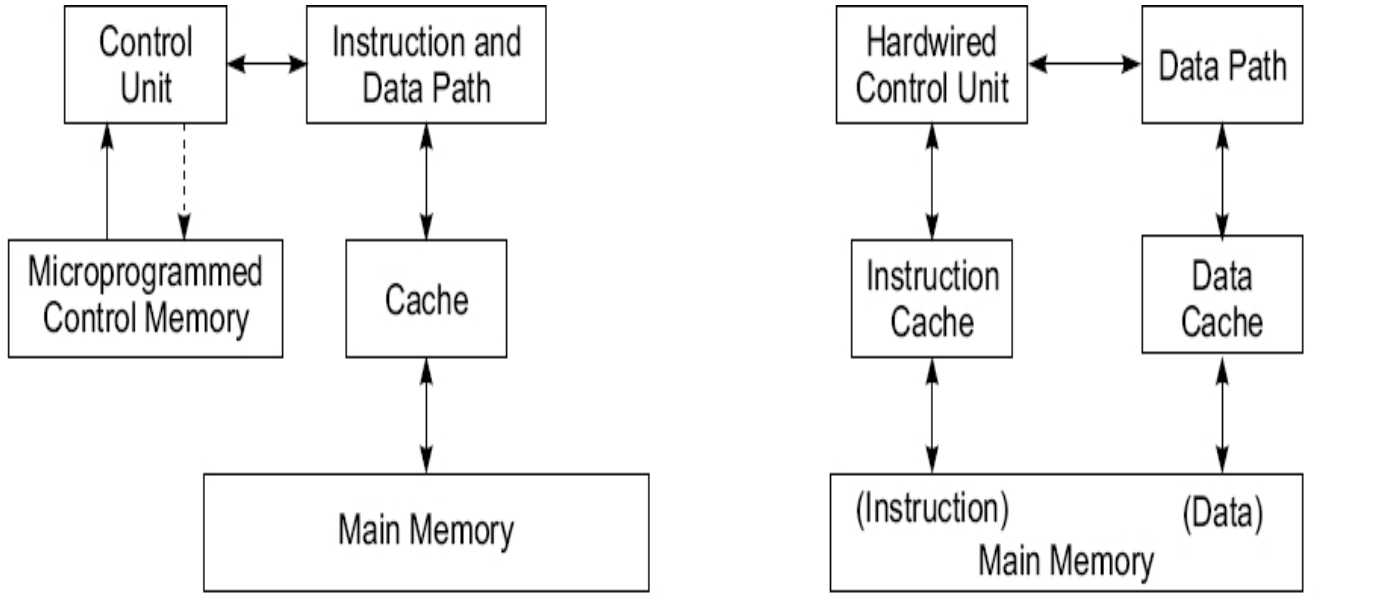
### Instruction Set Architectures

- **CISC**
  - **Many different instructions**
  - **Many different operand data types**
  - **Many different operand addressing formats**
  - **Relatively small number of general purpose registers**
  - **Many instructions directly match high-level language constructions**
- **RISC**
  - **Many fewer instructions than CISC (freeing chip space for more functional units!)**
  - **Fixed instruction format (e.g. 32 bits) and simple operand addressing**
  - **Relatively large number of registers**
  - **Small CPI (close to 1) and high clock rates**
-

## Instruction Set Architectures cond…

- **Architectural Distinctions**
- **CISC**
  - **Unified cache for instructions and data (in most cases)**
  - **Microprogrammed control units and ROM in earlier processors (hard-wired controls units now in some CISC systems)**
- **RISC**
  - **Separate instruction and data caches**
  - **Hard-wired control units**

## Instruction Set Architectures cond…

- 


(a) The CISC architecture with microprogrammed control and unified cache

(b) The RISC architecture with hardwired control and split instruction cache and data cache

**Fig. 4.4** Distinctions between typical RISC and typical CISC processor architectures (Courtesy of Gordon Bell, 1989)

## Instruction Set Architectures cond…

**Table 4.1** *Characteristics of Typical CISC and RISC Architectures*

| *Architectural Characteristic* | *Complex Instruction Set Computer (CISC)* | *Reduced Instruction Set Computer (RISC)* |
|---|---|---|
| Instruction-set size and instruction formats | Large set of instructions with variable formats (16–64 bits per instruction). | Small set of instructions with fixed (32-bit) format and most register-based instructions. |
| Addressing modes | 12–24. | Limited to 3–5. |
| General-purpose registers and cache design | 8–24 GPRs, originally with a unified cache for instructions and data, recent designs also use split caches. | Large numbers (32–192) of GPRs with mostly split data cache and instruction cache. |
| CPI | CPI between 2 and 15. | One cycle for almost all instructions and an average CPI < 1.5. |
| CPU Control | Earlier microcoded using control memory (ROM), but modern CISC also uses hardwired control. | Hardwired without control memory. |

## Instruction Set Architectures cond…

- **CISC Advantages**
  - Smaller program size (fewer instructions)
  - Simpler control unit design
  - Simpler compiler design
- 
- **RISC Advantages**
  - Has potential to be faster
  - Many more registers
- **RISC Problems**
  - More complicated register decoding system
  - Hardwired control is less flexible than microcode
- 
- 

## Instruction Set Architectures cond…

### ➢ CISC Scalar Processors

- Early systems had only integer fixed point facilities.
- Modern machines have both fixed and floating point facilities, sometimes as parallel functional units.
- Many CISC scalar machines are underpipelined.

### ➢ Representative CISC Processors:

- VAX 8600
- Motorola MC68040
- Intel Pentium

# Module 2
# Hardware Technologies

## Instruction Set Architectures cond…

- **VAX 8600 processor - CISC**

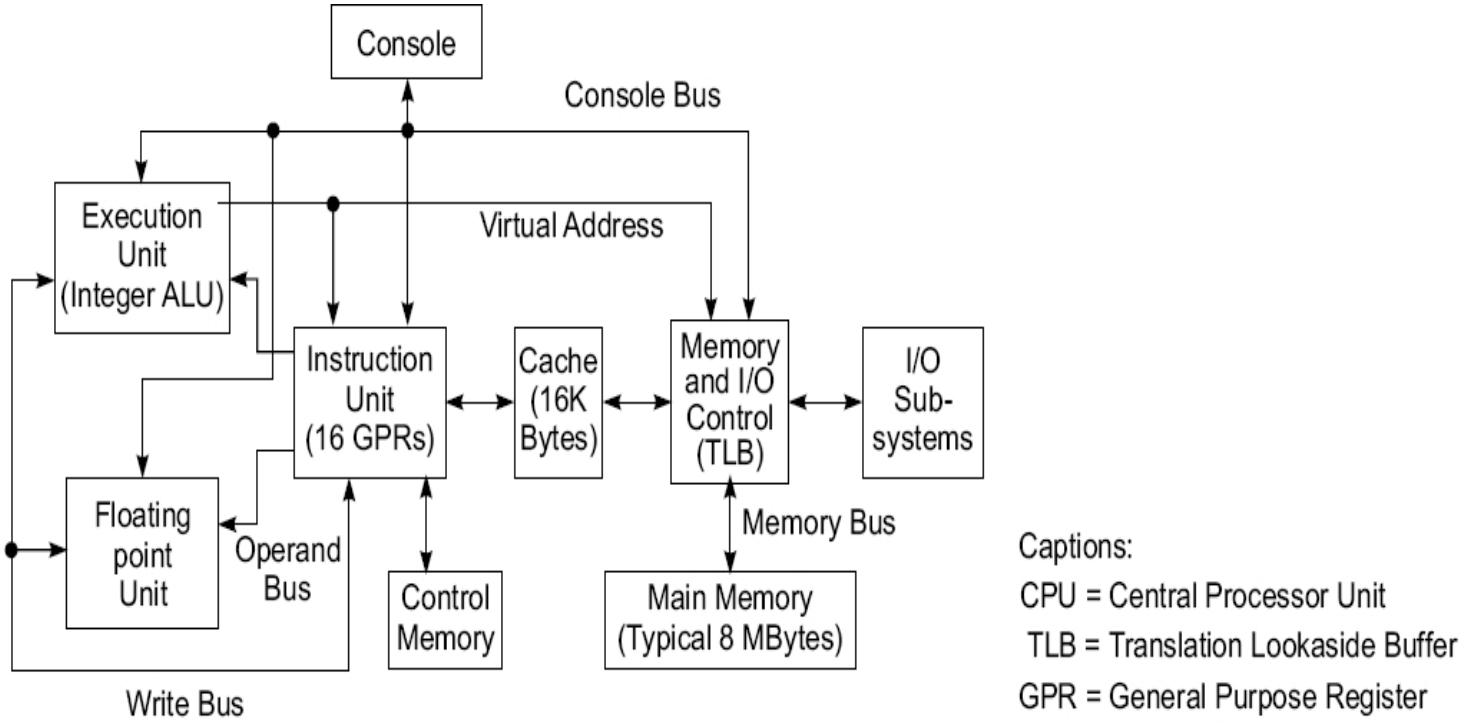



**Fig. 4.5** The VAX 8600 CPU, a typical CISC processor architecture (Courtesy of Digital Equipment Corporation, 1985)

## Instruction Set Architectures cond…VAX 8600 processor

- The VAX 8600 was introduced by Digital Equipment Corporation in 1985.
- This machine implemented a typical CISC architecture with microprogrammed control.
- The instruction set contained about 300 instructions with 20 different addressing modes.
- The CPU in the VAX 8600 consisted of two functional units for concurrent execution of integer and floating point instructions.
- The unified cache was used for holding both instructions and data.
- There were 16 GPRs in the instruction unit. Instruction pipelining was built with six stages in the VAX 8600, as in most else machines.
- The instruction unit prefetched and decoded instructions, handled branching operations, and supplied operands to the two functional units in a pipelined fashion.
- A Translation Lookaside Buffer (TLB) was used in the memory control unit for fast generation of a physical address from a virtual address.
- Both integer and floating point units were pipelined.
- The performance of the processor pipelines relied heavily on the cache hit ratio and on minimal branching damage to the pipeline flow.
-

# Module 2

## Hardware Technologies-RISC Scalar Processors

### RISC Scalar Processors

- Generic RISC processor are called scalar RISC because they are designed to issue one instruction per cycle.
- RISC and CISC scalar processors should have same performance if clock rate and program lengths are equal.
- RISC moves less frequent operations into software, thus dedicating hardware resources to the most frequently used operations.
- Representative RISC Processors:
  - Sun SPARC (Scalable Processor Architecture)
  - Intel i860
  - Motorola M88100
  - AMD 29000

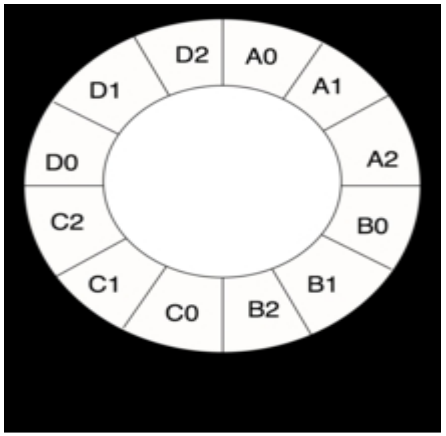
- All these processors use 32-bit instructions.
- The instruction set consists of 51 to 124 basic instructions.
- On-chip floating point units are built into the i860 and M88100.
- Off-chip floating point units are built into the SPARC and AMD.
- The scalability of SPARC architecture refers to the use of a different number of register windows in different SPARC implementation.

**Note:**

**register windows:** are a feature which dedicates registers to a subroutine by dynamically aliasing a subset of internal registers to fixed, programmer-visible registers. Register windows are implemented to improve the performance of a processor by reducing the number of stack operations required for function calls and returns.

**Register file:** The complete set of registers are known register file

**Register Window:** any particular set of registers in register file is know as a register window.




Example of a 4-window register window system

**In case of M88100 scalability reference to number of Special Functional Units (SFU) implementable on different versions of the M88100.**

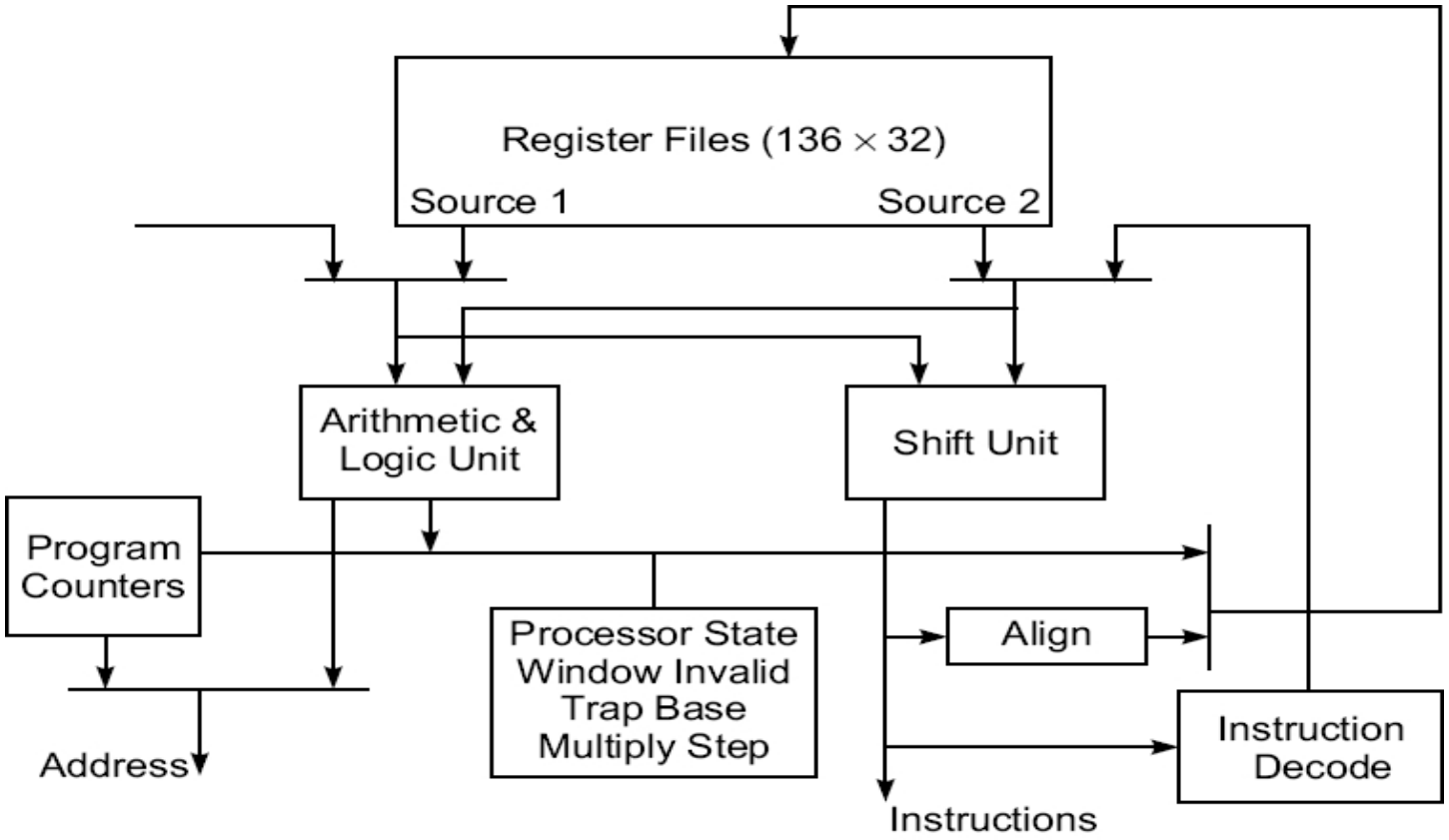
## SPARC architecture

**Different technologies and different window numbers are used by different SPARC manufactures.**

**All these manufactures implemented the Floating Point Unit (FPU) on a separate coprocessor chip. The SPARC architecture contains essentially a RISC Integer Unit (IU) implemented with 2 to 32 register windows.**

# Module 2
## Hardware Technologies-RISC Scalar Processors

## SPARCs (Scalable Processor Architecture) and Register Windows

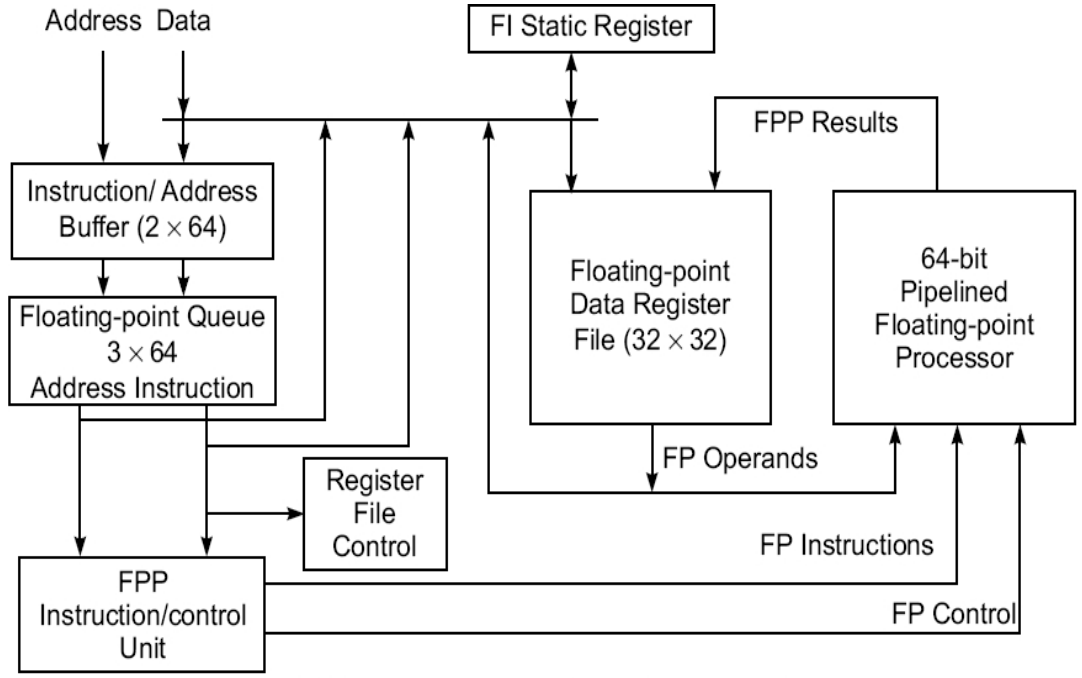
- 


(a) The Cypress CY7C601 SPARC processor

# Module 2
# Hardware Technologies

## Instruction Set Architectures cond…

## SPARCs (Scalable Processor Architecture) and Register Windows




(b) The Cypress CY7C602 floating-point unit

**Fig. 4.7** The SPARC architecture with the processor and the floating-point unit on two separate chips (Courtesy of Cypress Semiconductor Co., 1991)

## SPARCs and Register Windows

- SPARC family chips produced by Cypress Semiconductors, Inc. Figure 4.7 shows the architecture of the Cypress CY7C601 SPARC processor and of the CY7C602 FPU.
- The Sun SPARC instruction set contains 69 basic instructions
- The SPARC runs each procedure with a set of thirty-two 32-bit IU registers.
- Eight of these registers are global registers shared by all procedures, and the remaining 24 are window registers associated with only each procedure.
- The concept of using overlapped register windows is the most important feature introduced by the Berkeley RISC architecture.

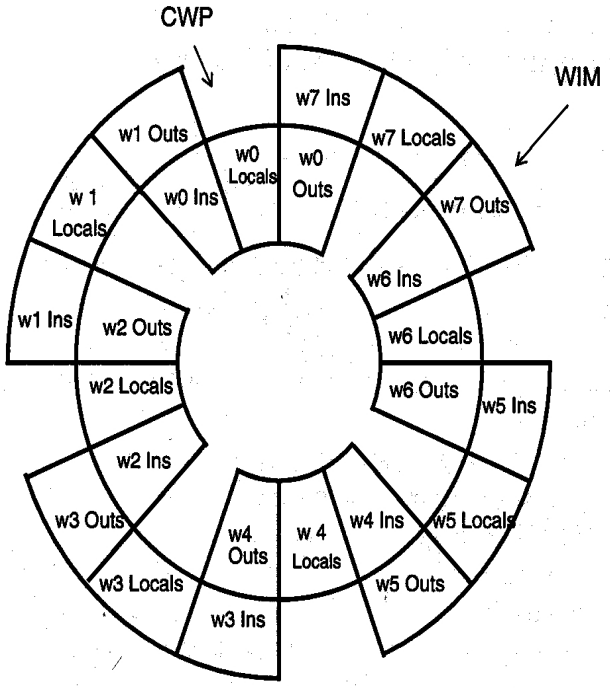
# Module 2
# Hardware Technologies

## Instruction Set Architectures cond…

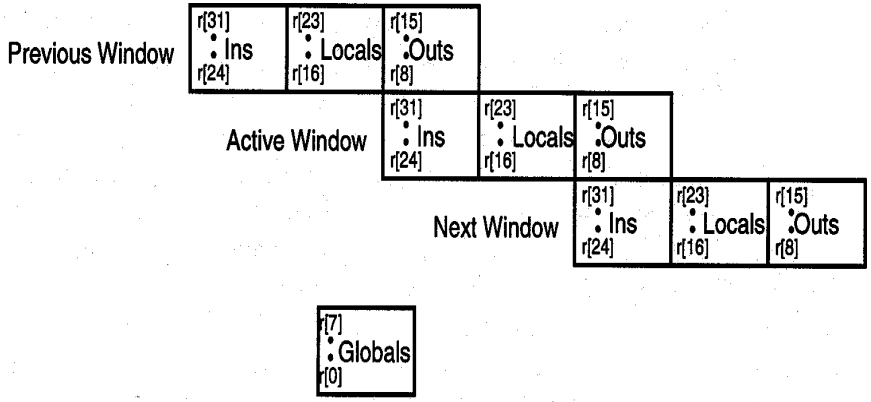
### SPARCs (Scalable Processor Architecture) and Register Windows

- Fig. 4.8 shows eight overlapping windows (formed with 64 local registers and 64 overlapped registers) and eight global with a total of 136 registers, as implemented in the Cypress 601.
- Each register window is divided into three eight-register sections, labeled Ins, Locals, and Outs.
- The local registers are only locally addressable by each procedure. The Ins and Outs are shared among procedures.
- The calling procedure passes parameters to the called procedure via its Outs (r8 to r15) registers, which are the Ins registers of the called procedure.
- The window of the currently running procedure is called the active window pointed to by a current window pointer.
- A window invalid mask is used to indicate which window is invalid. The trap base register serves as a pointer to a trap handler.

# Module 2
## Hardware Technologies




(b) Eight register windows forming a circular stack




(a) Three overlapping register windows and the global registers

**Figure 4.8 The concept of overlapping register windows in the SPARC architecture.** (Courtesy of Sun Microsystems, Inc., 1987)

# Module 2

## Hardware Technologies-Superscalar, Vector Processors

## Superscalar and Vector Processors

(a vector processor or array processor is a central processing unit (CPU) that implements an instruction set where its instructions are designed to operate efficiently and effectively on large one-dimensional arrays of data called vectors.)

- A CISC or a RISC scalar processor can be improved with a superscalar or vector architecture.
- Scalar processors are those executing one instruction per cycle.
- Only one instruction is issued per cycle, and only one completion of instruction is expected from the pipeline per cycle.
- In a superscalar processor, multiple instructions are issued per cycle and multiple results are generated per cycle.
- A vector processor executes vector instructions on arrays of data; each vector instruction involves a string of repeated operations, which are ideal for pipelining with one result per cycle.

(Vector instructions are a class of instructions that enable parallel processing of data sets. An entire array of integers or floating point numbers is processed in a single operation, eliminating the loop control mechanism typically found in processing arrays.)

# Superscalar Processors

- Superscalar processors are designed to exploit more instruction-level parallelism in user programs.
- Only independent instructions can be executed in parallel without causing a wait state. The amount of instruction level parallelism varies widely depending on the type of code being executed.
- It has been observed that the average value is around 2 for code without loop unrolling. Therefore, for these codes there is not much benefit gained from building a machine that can issue more than three instructions per cycle.
- The instruction-issue degree in a superscalar processor has thus been limited to 2 to 5 in practice.

# Module 2

## Hardware Technologies-Superscalar Processors

## Pipelining in Superscalar Processors

- **The fundamental structure of a three-issue superscalar pipeline is illustrated in Fig. 4.11.**

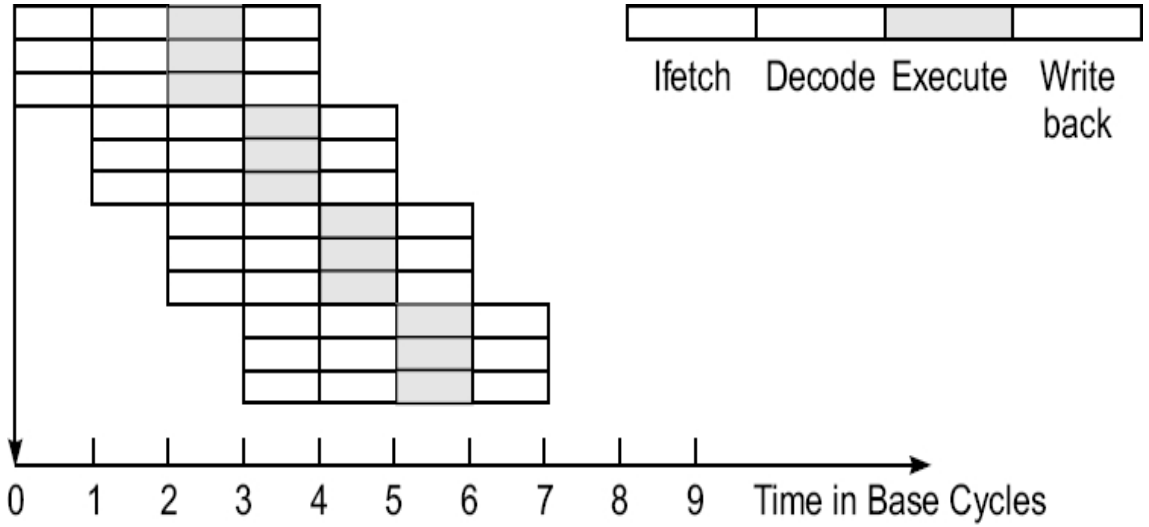



**Fig. 4.11** A superscalar processor of degree m = 3

# Pipelining in Superscalar Processors

- A superscalar processor of degree m can issue m instructions per cycle.
- The base scalar processor, implemented either in RISC or CISC, has m = 1.
- In order to fully utilize a superscalar processor of degree m, m instructions must be executable in parallel. This situation may not be true in all clock cycles.
- In that case, some of the pipelines may be stalling in a wait state.
- In a superscalar processor, the simple operation latency should require only one cycle, as in the base scalar processor.
- Due to the desire for a higher degree of instruction-level parallelism in programs, the superscalar processor depends more on an optimizing compiler to exploit parallelism.

## Representative Superscalar Processors

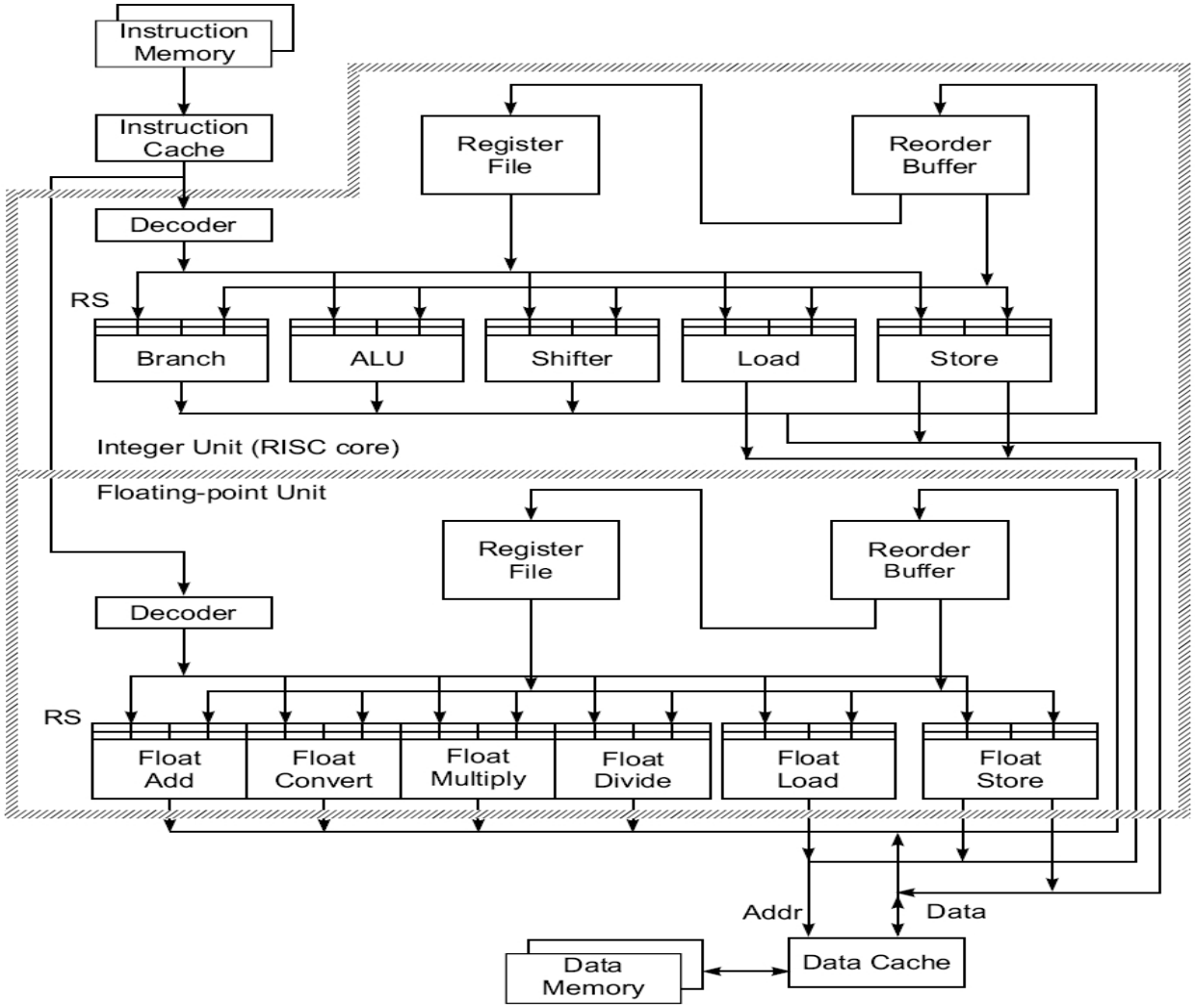



**Fig. 4.12** A typical superscalar RISC processor architecture consisting of an integer unit and a floating-point unit (Courtesy of M. Johnson, 1991; reprinted with permission from Prentice-Hall, Inc.)

Reservation stations **permit the CPU to fetch and re-use** a data value as soon as it has been computed, rather than waiting for it to be stored in a register and re-read. When instructions are issued, they can designate the reservation station from which they want their input to read.

# Representative Superscalar Processors

- Typical Superscalar Architecture will have
  - multiple instruction pipelines
  - an instruction cache that can provide multiple instructions per fetch
  - multiple buses among the function units
- The maximum number of instructions issued per cycle ranges from two to five in these superscalar processors.
- Typically register files in the IU and FPU each have 32 registers.
- Most superscalar processor implement the IU and FPU on same chip.

# VLIW Architecture (VLIW-Very Long Instruction Word)

- A typical VLIW machine has instruction words hundreds of bits in length

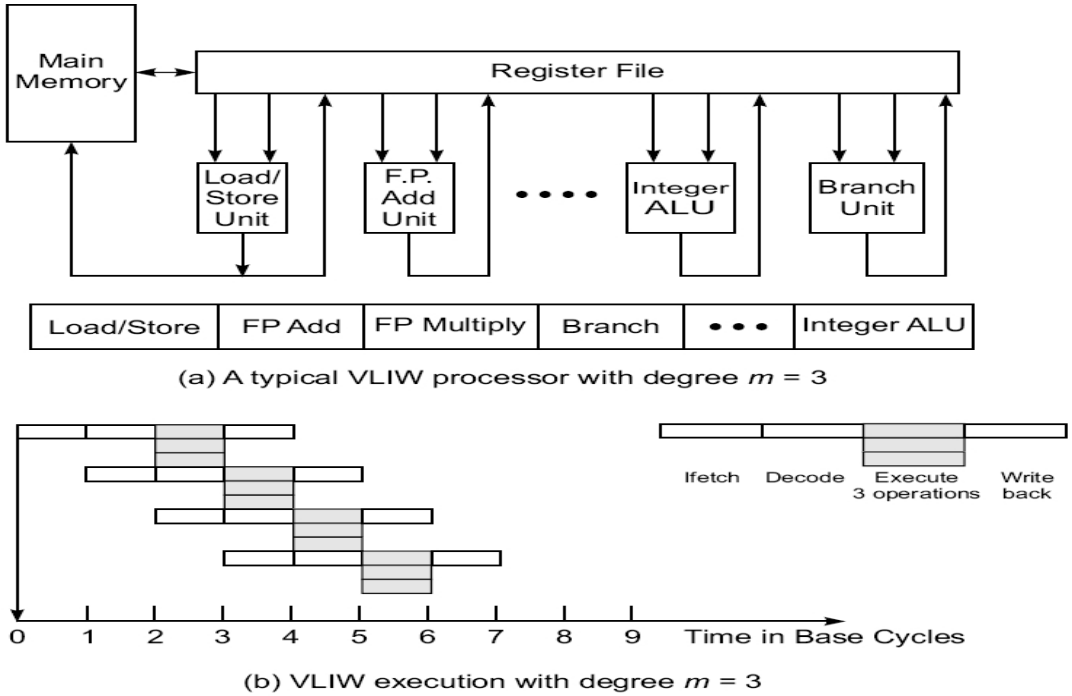



**Fig. 4.14** The architecture of a very long instruction word (VLIW) processor and its pipeline operations (Courtesy of Multiflow Computer, Inc., 1987)

# VLIW Architecture

- Instructions usually hundreds of bits long.
- Each instruction word essentially carries multiple short instructions.

( Each instruction VLIW specifies multiple operations. )

- Each of the short instructions are effectively issued at the same time.
- (This is related to the long words frequently used in microcode.)
- Compilers for VLIW architectures should optimally try to predict branch outcomes to properly group the instructions.
- Multiple functional units are used concurrently in VLIW processor.
- All functional units share the use of a common large register file.
- The operations to be simultaneously executed by the functional units are synchronized in VLIW instructions.

# Module 2
# Hardware Technologies-VLIW Architecture

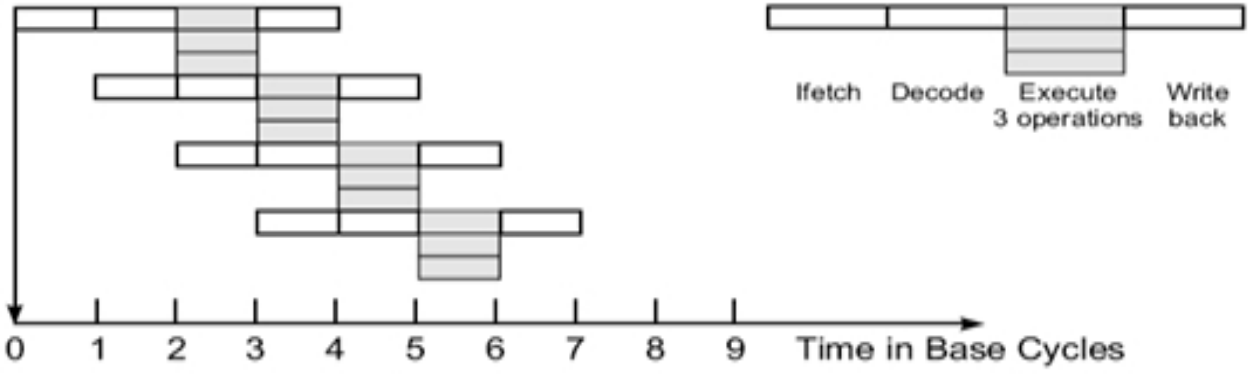
## Differences between VLIW and superscalar processor

**1. Decoding of instructions is easier in VLIW than in superscalar**

**2. Code density in VLIW is less than in superscalar's, because if a region of a VLIW word isn't needed in a particular instruction, it must still exist (to be filled with a —no op).**

**3. Superscalar can be compatible with scalar processors this is difficult with VLIW parallel and non-parallel architectures.**

## Pipelining in VLIW Processors




(b) VLIW execution with degree $m = 3$

## VLIW Opportunities

- Random parallelism among scalar operations is exploited in VLIW, instead of regular parallelism in a vector or SIMD machine.
- The efficiency of the machine is entirely dictated by the success, or goodness, of the compiler in planning the operations to be placed in the same instruction words.
- Different implementations of the same VLIW architecture may not be binary-compatible with each other, resulting in different latencies.

## VLIW Summary

- VLIW reduces the effort required to detect parallelism using hardware or software techniques.
- The main advantage of VLIW architecture is its simplicity in hardware structure and instruction set.
- Unfortunately, VLIW does require careful analysis of code in order to —compact the most appropriate short instructions into a VLIW word.

# Module 2

## Hardware Technologies-Vector Processors

**Vector Processors (a vector processor or array processor is a central processing unit (CPU) that implements an instruction set where its instructions are designed to operate efficiently and effectively on large one-dimensional arrays of data called vectors.)**

- **A vector processor is a coprocessor designed to perform vector computations.**
- **A vector is a one-dimensional array of data items (each of the same data type).**
- **Vector processors are often used in multipipelined supercomputers.**
- **Architectural types include:**
    1. **Register-to-Register (with shorter instructions and register files)**
    2. **Memory-to-Memory (longer instructions with memory addresses)**

## 1. Register-to-Register Vector Instructions

Let

- Vi : a vector register of length n,
- si : a scalar register,
- M(1:n): a memory array of length n,
- o : is a vector operation.
- Typical instructions include the following:

  - V1 o V2 →V3 (element by element operation)
  - s1 o V1 → V2 (scaling of each element)
  - V1 o V2 → s1 (binary reduction - i.e. sum of products)
  - M(1:n) → V1 (load a vector register from memory)
  - V1 → M(1:n) (store a vector register into memory)
  - o V1 → V2 (unary vector -- i.e. negation)
  - o V1 → s1 (unary reduction -- i.e. sum of vector)

## 2. Memory-to-Memory Vector Instructions

- Typical memory-to-memory vector instructions (using the same notation as given in the previous slide) include these:
  - M1(1:n) o M2(1:n) → M3(1: n) (binary vector)
  - s1 o M1(1:n) → M2(1:n) (scaling)
  - o M1(1:n) → M2(1:n) (unary vector)
  - M1(1:n) o M2(1:n) → M(k) (binary reduction)

# Module 2

## Hardware Technologies-Vector Processors

## Pipelines in Vector Processors

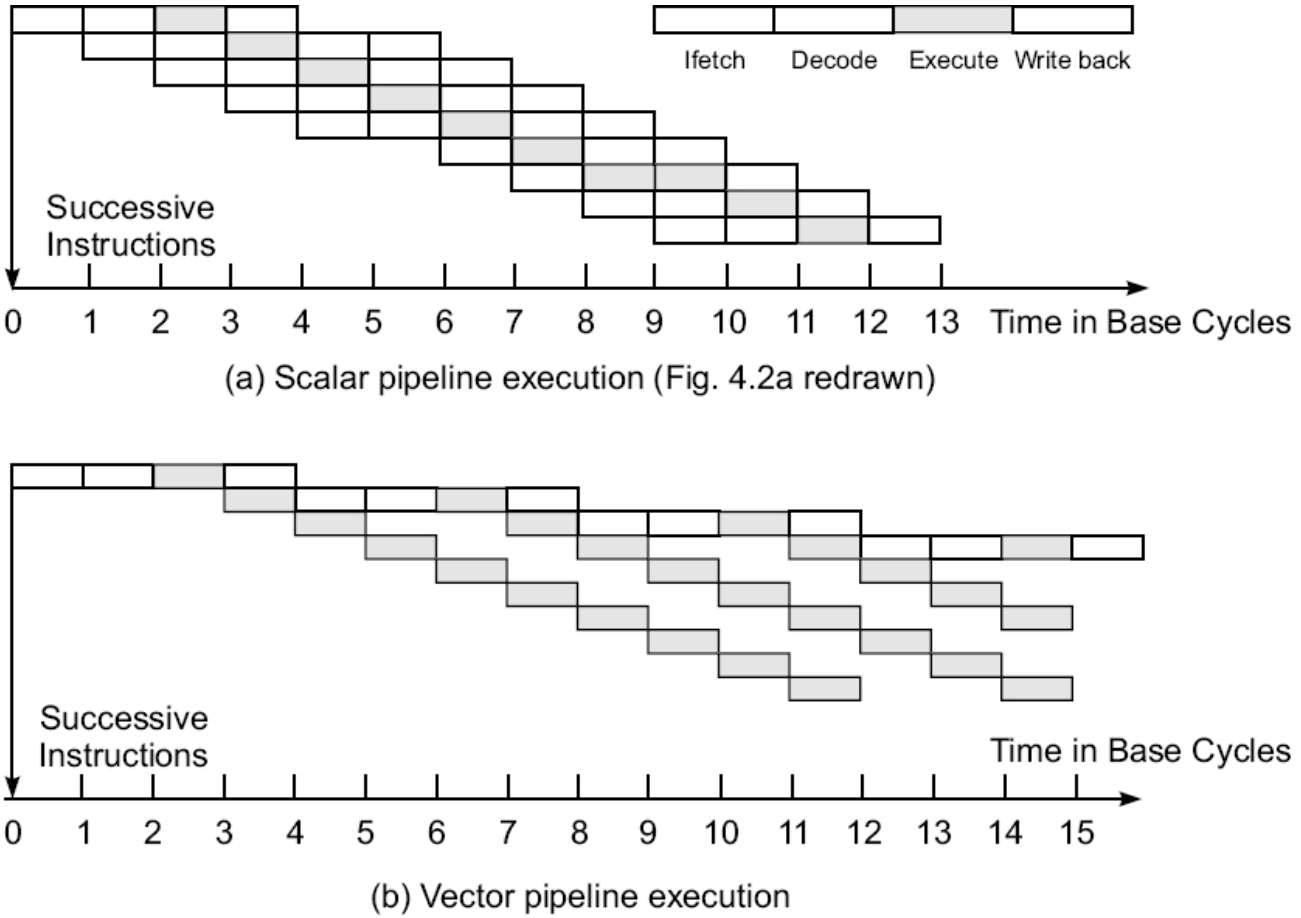



**Fig. 4.15** Pipelined execution in a base scalar processor and in a vector processor, respectively (Courtesy of Jouppi and Wall; reprinted from Proc. ASPLOS, ACM Press, 1989)

# Module 2

## Hardware Technologies-Vector Processors

- **Vector processors can usually effectively use large pipelines in parallel, the number of such parallel pipelines effectively limited by the number of functional units.**
- **As usual, the effectiveness of a pipelined system depends on the availability and use of an effective compiler to generate code that makes good use of the pipeline facilities.**

## Symbolic Processors

- Symbolic processors are somewhat unique in that their architectures are tailored toward the execution of programs in languages similar to LISP, Scheme, and Prolog.
- In effect, the hardware provides a facility for the manipulation of the relevant data objects with —tailored instructions.
- These processors (and programs of these types) may invalidate assumptions made about more traditional scientific and business computations.

# Symbolic Processors

**Table 4.6** *Characteristics of Symbolic Processing*

| *Attributes* | *Characteristics* |
|---|---|
| Knowledge Representations | Lists, relational databases, scripts, semantic nets, frames, blackboards, objects, production systems. |
| Common Operations | Search, sort, pattern matching, filtering, contexts, partitions, transitive closures, unification, text retrieval, set operations, reasoning. |
| Memory Requirements | Large memory with intensive access pattern. Addressing is often content -based. Locality of reference may not hold. |
| Communication Patterns | Message traffic varies in size and destination; granularity and format of message units change with applications. |
| Properties of Algorithms | Nondeterministic, possibly parallel and distributed computations. Data dependences may be global and irregular in pattern and granularity. |
| Input/Output requirements | User-guided programs; intelligent person-machine interfaces; inputs can be graphical and audio as well as from keyboard; access to very large on-line databases. |
| Architecture Features | Parallel update of large knowledge bases, dynamic load balancing; dynamic memory allocation; hardware-supported garbage collection; stack processor architecture; symbolic processors. |

# Module 2

## Hardware Technologies-Memory Hierarchical Technology

### Memory Hierarchical Technology

- **Storage devices such as registers, caches, main memory, disk devices, and backup storage are often organized as a hierarchy as depicted in Fig. 4.17.**

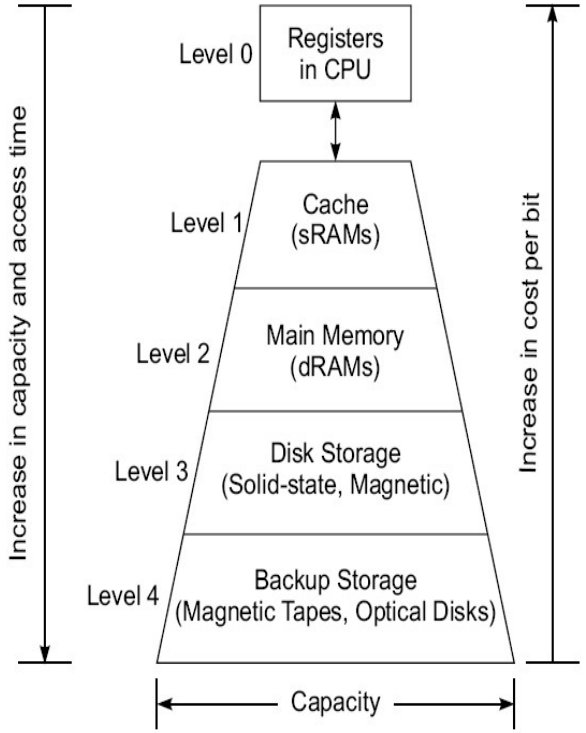



**Fig. 4.17** A four-level memory hierarchy with increasing capacity and decreasing speed and cost from low to high levels

## Memory Hierarchical Technology

- The memory technology and storage organization at each level is characterized by five parameters:
- Access Time - ti (round-trip time from CPU to ith level)
- Memory Size - si (number of bytes or words in level i)
- Cost Per Byte - ci
- Transfer Bandwidth - bi (rate of transfer between levels)
- Unit of Transfer - xi (grain size for transfers between levels i and i+1)

## Memory Hierarchical Technology

- **Memory devices at a lower level are:**
  - **faster to access,**
  - **are smaller in capacity,**
  - **are more expensive per byte,**
  - **have a higher bandwidth, and**
  - **have a smaller unit of transfer.**
- **In general, $t_{i-1} < t_i$, $s_{i-1} < s_i$, $c_{i-1} > c_i$, $b_{i-1} > b_i$ and $x_{i-1} < x_i$ for i = 1, 2, 3, and 4 in the hierarchy where i = 0 corresponds to the CPU register level.**
- **The cache is at level 1, main memory at level 2, the disks at level 3 and backup storage at level 4.**

# Module 2

## Hardware Technologies-Memory Hierarchical Technology

## Memory Hierarchical Technology

### – Registers and Caches Registers

- The registers are parts of the processor;
- Register assignment is made by the compiler.
- Register transfer operations are directly controlled by the processor after instructions are decoded.
- Register transfer is conducted at processor speed, in one clock cycle.

### – Caches

- The cache is controlled by the MMU (Memory Management Unit ) and is programmer-transparent.
- The cache can also be implemented at one or multiple levels, depending on the speed and application requirements.
- Multi-level caches are built either on the processor chip or on the processor board.
- Multi-level cache systems have become essential to deal with memory access latency.

### – Main Memory (Primary Memory)

- It is usually much larger than the cache and often implemented by the most cost-effective RAM chips, such as DDR SDRAMs (Dual Data Rate Synchronous Dynamic RAMs)
- The main memory is managed by a MMU in cooperation with the operating system.

# Module 2

## Hardware Technologies-Memory Hierarchical Technology

## Memory Hierarchical Technology

### – Disk Drives and Backup Storage

- The disk storage is considered the highest level of on-line memory.
- It holds the system programs such as the OS and compilers, and user programs and their data sets.
- Optical disks and magnetic tape units are off-line memory for use as archival and backup storage.
- They hold copies of present and past user programs and processed results and files.
- Disk drives are also available in the form of RAID arrays.

### – Peripheral Technology

- Peripheral devices include printers, plotters, terminals, monitors, graphics displays, optical scanners, image digitizers, output microfilm devices etc.
- Some I/O devices are tied to special-purpose or multimedia applications.

# Module 2

## Hardware Technologies-Memory Hierarchical Technology

## Memory Hierarchical Technology

### – Disk Drives and Backup Storage

- The disk storage is considered the highest level of on-line memory.
- It holds the system programs such as the OS and compilers, and user programs and their data sets.
- Optical disks and magnetic tape units are off-line memory for use as archival and backup storage.
- They hold copies of present and past user programs and processed results and files.
- Disk drives are also available in the form of RAID (Redundant Arrays of Independent Disks) arrays.

### – Peripheral Technology

- Peripheral devices include printers, plotters, terminals, monitors, graphics displays, optical scanners, image digitizers, output microfilm devices etc.
- Some I/O devices are tied to special-purpose or multimedia applications.

# Inclusion, Coherence, and Locality

- Information stored in a memory hierarchy (M1, M2,…, Mn) satisfies 3 important properties:
  - Inclusion
  - Coherence
  - Locality
- We consider cache memory the innermost level M1, which directly communicates with the CPU registers.
- The outermost level Mn contains all the information words stored. In fact, the collection of all addressable words in Mn forms the virtual address space of a computer.
- Program and data locality is characterized below as the foundation for using a memory hierarchy effectively.

## Inclusion, Coherence, and Locality

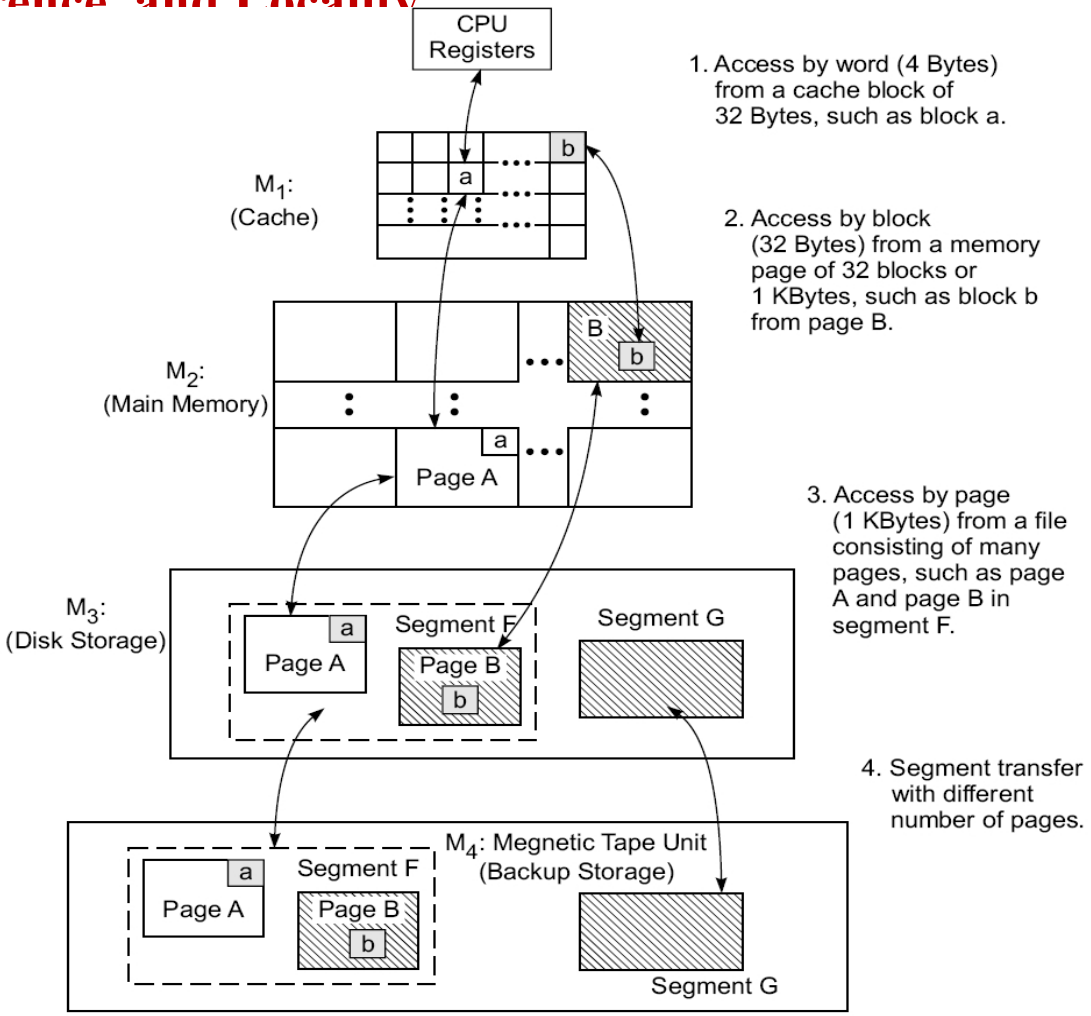



**Fig. 4.18** The inclusion property and data transfers between adjacent levels of a memory hierarchy

# Inclusion, Coherence, and Locality

## 1. The Inclusion Property

- **The inclusion property is stated as:**

  **M1 ⊂ M2 ⊂ ... ⊂ Mn**

- **The implication of the inclusion property is that all items of information in the —innermost memory level (cache) also appear in the outer memory levels.**
- **The inverse, however, is not necessarily true. That is, the presence of a data item in level Mi+1 does not imply its presence in level Mi. We call a reference to a missing item —miss.**

# Inclusion, Coherence, and Locality

## 2. The Coherence Property

- The requirement that copies of data items at successive memory levels be consistent is called the **Coherence Strategies**
  - **Write-through**
    - As soon as a data item in Mi is modified, immediate update of the corresponding data item(s) in Mi+1, Mi+2, … Mn is required.
    - This is the most aggressive (and expensive) strategy.
  - **Write-back**
- The update of the data item in Mi+1 corresponding to a modified item in Mi is not updated untill it (or the block/page/etc. in Mi that contains it) is replaced or removed.
- This is the most efficient approach, but cannot be used (without modification) when multiple processors share Mi+1, …, Mn.

# Inclusion, Coherence, and Locality

## 3. Locality of References

- Memory references are generated by the CPU for either instruction or data access.
- These accesses tend to be clustered in certain regions in time, space, and ordering. There are three dimensions of the locality property:
  - Temporal locality – if location M is referenced at time t, then it (location M) will be referenced again at some time t+Dt.
  - Spatial locality – if location M is referenced at time t, then another location M±Dm will be referenced at time t+Dt.
  - Sequential locality – if location M is referenced at time t, then locations M+1, M+2, … will be referenced at time t+Dt, t+Dt', etc.
- In each of these patterns, both Dm and Dt are —small.
- H&P suggest that 90 percent of the execution time in most programs is spent executing only 10 percent of the code.

# Working Sets

- The set of addresses (bytes, pages, etc.) referenced by a program during the interval from t to t+Dt, where Dt is called the working set parameter, changes slowly.
- This set of addresses, called the working set, should be present in the higher levels of M if a program is to execute efficiently (that is, without requiring numerous movements of data items from lower levels of M). This is called the working set principle.

# Working Sets

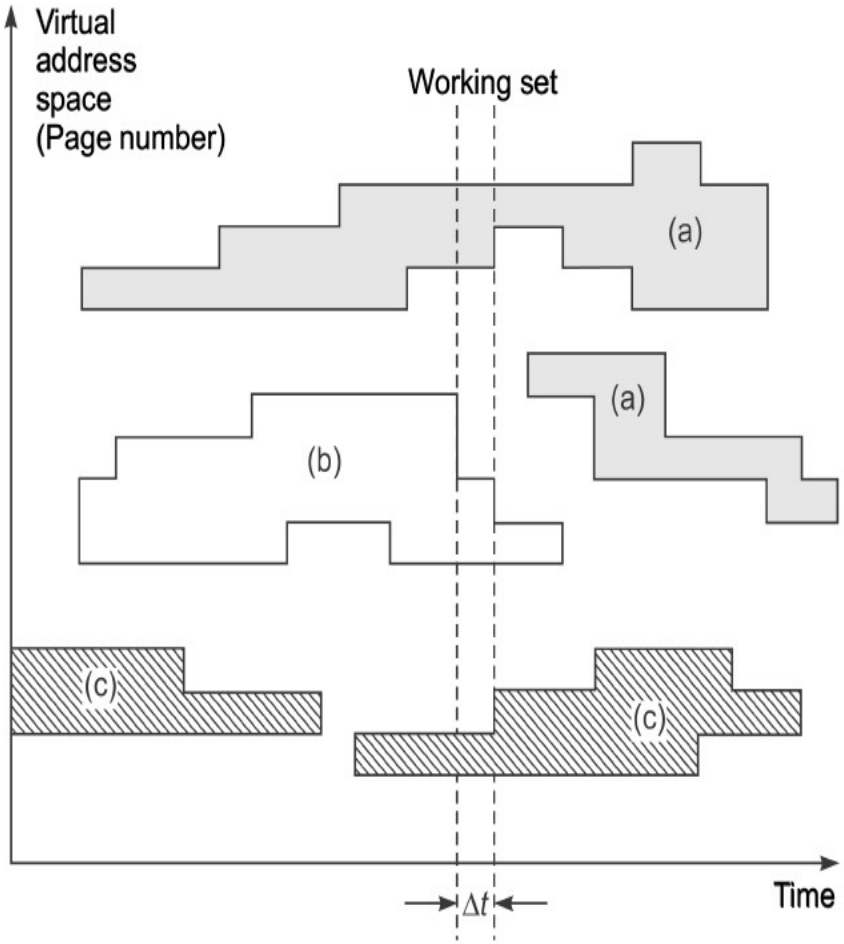



**Fig. 4.19** Memory reference patterns in typical program trace experiments, where regions (a), (b), and (c) are generated with the execution of three software processes

# Memory Capacity Planning

- The performance of a memory hierarchy is determined by the Effective Access Time $T_{eff}$ to any level in the hierarchy. It depends on the hit ratios and access frequencies.

## 1. Hit Ratios

- When a needed item (instruction or data) is found in the level of the memory hierarchy being examined, it is called a hit. Otherwise (when it is not found), it is called a miss (and the item must be obtained from a lower level in the hierarchy).
- The hit ratio, h, for Mi is the probability (between 0 and 1) that a needed data item is found when sought in level memory Mi.
- The miss ratio is obviously just 1-hi.

  We assume h0 = 0 and hn = 1.

# Memory Capacity Planning

## 2. Access Frequencies

- The access frequency fi to level Mi is

  $f_i = (1-h1) \times (1-h2) \times \ldots \times h_i$

  Note that $f1 = h1$, and $\sum_{i=1}^{n} f_i = 1$

## 3. Effective Access Times

- There are different penalties associated with misses at different levels in the memory hierarchy.
  - A cache miss is typically 2 to 4 times as expensive as a cache hit (assuming success at the next level).
  - A page fault (miss) is 3 to 4 magnitudes as costly as a page hit.

- The effective access time of a memory hierarchy can be expressed as

# Virtual Memory Technology

- To facilitate the use of memory hierarchies, the memory addresses normally generated by modern processors executing application programs are not physical addresses, but are rather virtual addresses of data items and instructions.
- Physical addresses, of course, are used to reference the available locations in the real physical memory of a system.
- Virtual addresses must be mapped to physical addresses before they can be used.

# Virtual to Physical Mapping

- The mapping from virtual to physical addresses can be formally defined as follows:

$$f_t v = \begin{cases} m, & \text{if } m \in M \text{ has been allocated to store} \\ & \text{the data identified by virtual address } v \\ \varnothing & \text{if data } v \textit{ is} \text{ missing in } M \end{cases}$$

- 
- The mapping returns a physical address if a memory hit occurs. If there is a memory miss, the referenced item has not yet been brought into primary memory.

- **Mapping Efficiency**
- **The efficiency with which the virtual to physical mapping can be accomplished significantly affects the performance of the system.**
- **Efficient implementations are more difficult in multiprocessor systems where additional problems such as coherence, protection, and consistency must be addressed.**
- **Virtual Memory Models**
  - **Private Virtual Memory**
  - **Shared Virtual Memory**

- **Virtual Memory Models**
  - **Private Virtual Memory**
  - **Shared Virtual Memory**

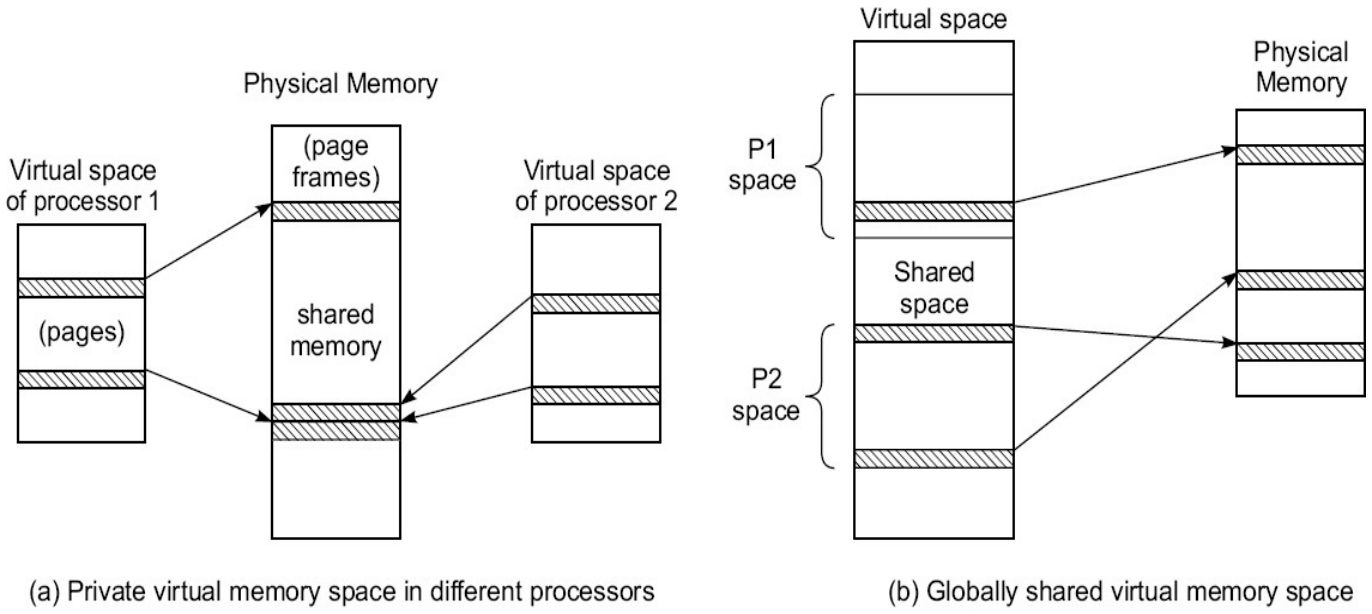



**Fig. 4.20** Two virtual memory models for multiprocessor systems (Courtesy of Dubois and Briggs, tutorial, *Annual Symposium on Computer Architecture*, 1990)

## – Private Virtual Memory

- **In this scheme, each processor has a separate virtual address space, but all processors share the same physical address space.**
  - **Advantages:**
    - **Small processor address space**
    - **Protection on a per-page or per-process basis**
    - **Private memory maps, which require no locking**
  - **Disadvantages**
    - **The synonym problem – different virtual addresses in different/same virtual spaces point to the same physical page**
    - **The same virtual address in different virtual spaces may point to different pages in physical memory**

## – Private Virtual Memory

- **In this scheme, each processor has a separate virtual address space, but all processors share the same physical address space.**
  - **Advantages:**
    - **Small processor address space**
    - **Protection on a per-page or per-process basis**
    - **Private memory maps, which require no locking**
  - **Disadvantages**
    - **The synonym problem – different virtual addresses in different/same virtual spaces point to the same physical page**
    - **The same virtual address in different virtual spaces may point to different pages in physical memory**

## Shared Virtual Memory

- All processors share a single shared virtual address space, with each processor being given a portion of it.
- Some of the virtual addresses can be shared by multiple processors.

### Advantages:

- All addresses are unique
- Synonyms are not allowed

### Disadvantages

- Processors must be capable of generating large virtual addresses (usually > 32 bits)
- Since the page table is shared, mutual exclusion must be used to guarantee atomic updates
- Segmentation must be used to confine each process to its own address space
- The address translation process is slower than with private (per processor) virtual memory

### Memory Allocation

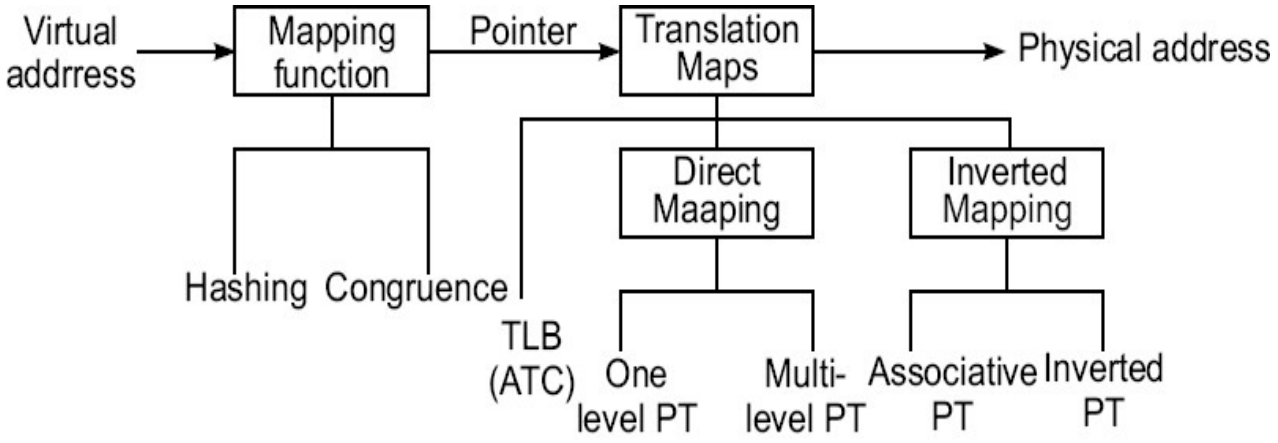
- Both the virtual address space and the physical address space are divided into fixed-length pieces.
  - In the virtual address space these pieces are called pages.
  - In the physical address space they are called page frames.
- The purpose of memory allocation is to allocate pages of virtual memory using the page frames of physical memory.

# TLB, Paging, and Segmentation

- Both the virtual memory and physical memory are partitioned into fixed-length pages. The purpose of memory allocation is to allocate pages of virtual memory to the page frames of the physical memory

## Address Translation Mechanisms

- The process demands the translation of virtual addresses into physical addresses. Various schemes for virtual address translation are summarized in Fig. 4.21a.




(a) Virtual addrress translation schemes (PT = page table)

# TLB, Paging, and Segmentation

- The translation demands the use of translation maps which can be implemented in various ways.
- Translation maps are stored in the cache, in associative memory, or in the main memory.
- To access these maps, a mapping function is applied to the virtual address. This function generates a pointer to the desired translation map.
- This mapping can be implemented with a hashing or congruence function.
- Hashing is a simple computer technique for converting a long page number into a short one with fewer bits.
- The hashing function should randomize the virtual page number and produce a unique hashed number to be used as the pointer.

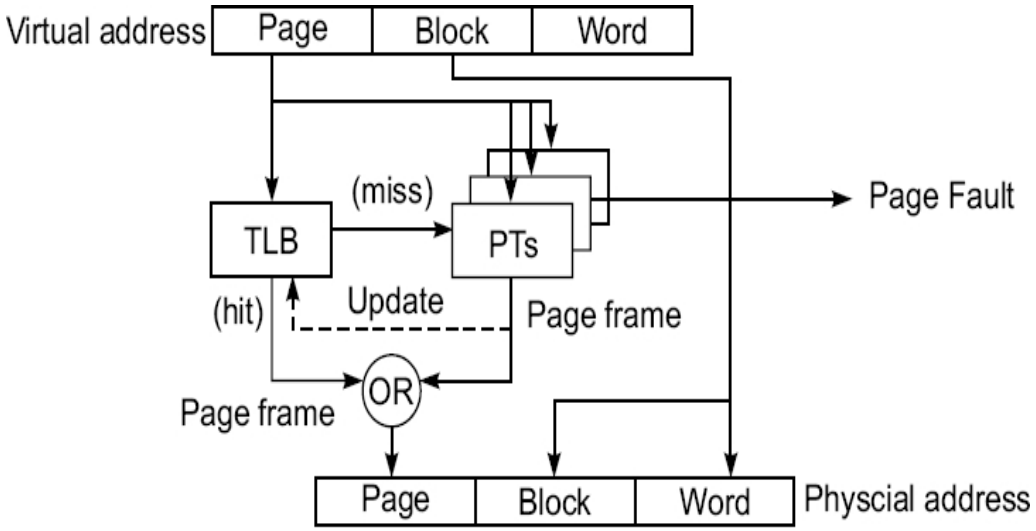
# TLB, Paging, and Segmentation

## Translation Lookaside Buffer

- The TLB is a high-speed lookup table which stores the most recently or likely referenced page entries.
- A page entry consists of essentially a (virtual page number, page frame number) pair. It is hoped that pages belonging to the same working set will be directly translated using the TLB entries.
- The use of a TLB and PTs for address translation is shown in Fig 4.21b. Each virtual address is divided into 3 fields:
  - The leftmost field holds the virtual page number,
  - the middle field identifies the cache block number,
  - the rightmost field is the word address within the block.

# TLB, Paging, and Segmentation

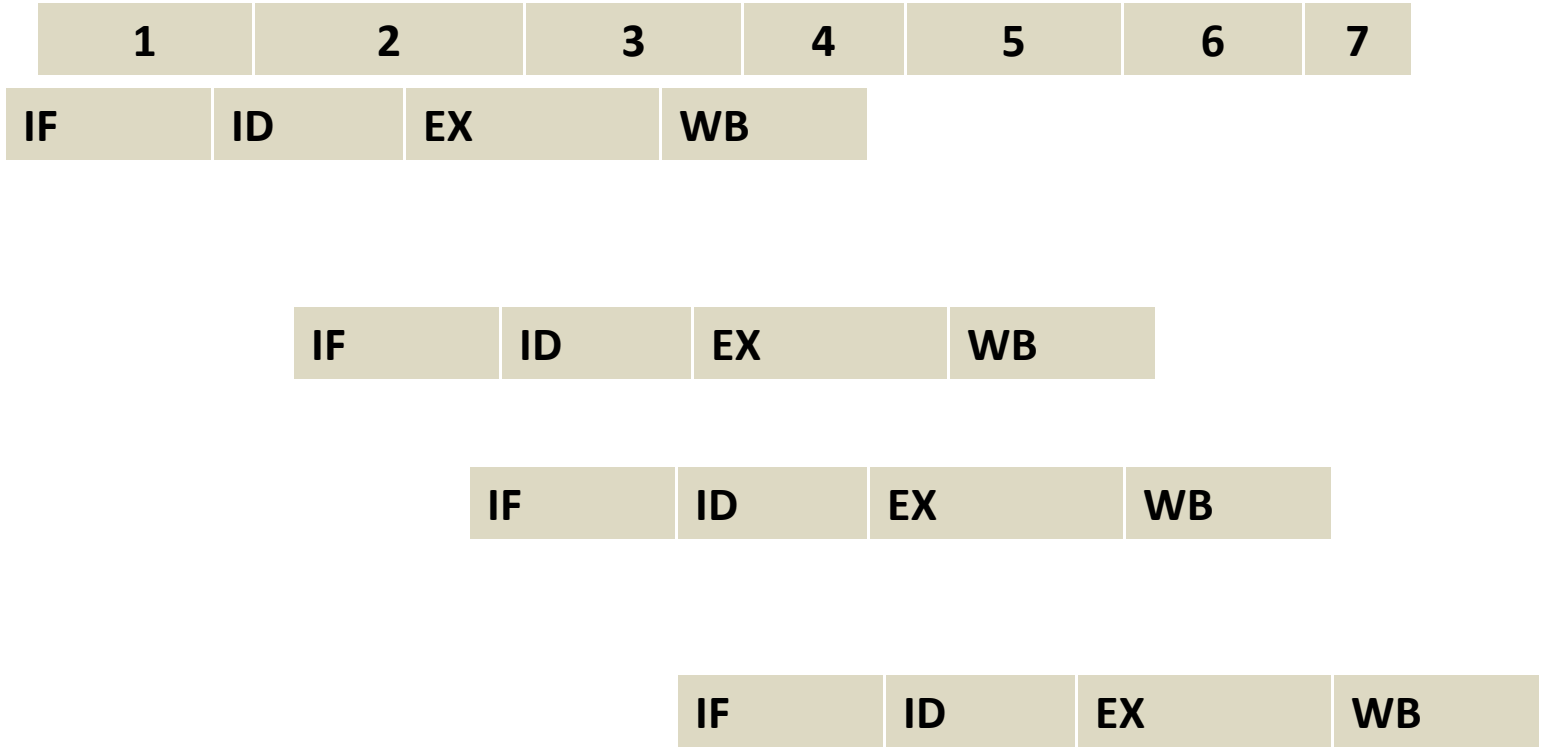
## Translation Lookaside Buffer

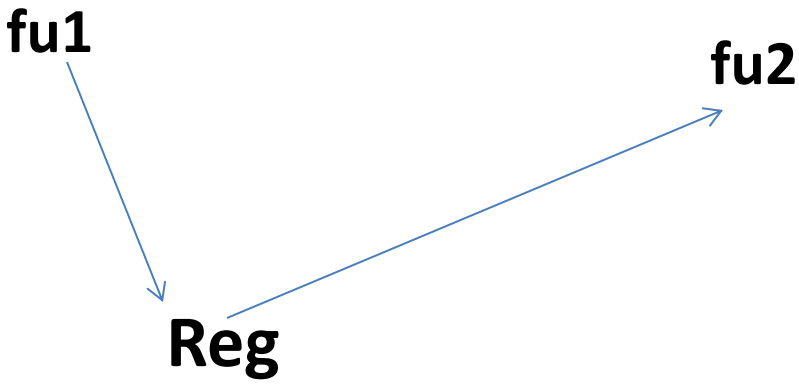



(b) Use of a TLB and PTs for address translation

# Translation Lookaside Buffer

- Our purpose is to produce the physical address consisting of the page frame number, the block number, and the word address.
- The first step of the translation is to use the virtual page number as a key to search through the TLB for a match.
- The TLB can be implemented with a special associative memory (content addressable memory) or use part of the cache memory.
- In case of a match (a hit) in the TLB, the page frame number is retrieved from the matched page entry. The cache block and word address are copied directly.
- In case the match cannot be found (a miss) in the TLB, a hashed pointer is used to identify one of the page tables where the desired page frame number can be retrieved.


1
2
3
4
5
6
7
IF
ID
EX
WB
IF
ID
EX
WB
IF
ID
EX
WB
IF
ID
EX
WB


fu1
fu2
Reg