SRN | | | | | | | | | | | |

PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

MODEL END SEMESTER ASSESSMENT B.TECH IV SEMESTER- May 2020

UE18CS253-Microprocessor and Computer Architecture

1.	a)	Differentiate between CISC and RISC type of Microprocessors.	10 Marks
	b)	Explain briefly RISC Instruction Set-Operations.	10 Marks

2.	a)	What are the major hurdles of pipelining? Explain any one of the hazards in detail.	10 Marks
	b)	With data path explain classic five stage pipeline for a RISC processor	10 Marks

3.	a)	Briefly explain six basic cache optimization schemes.	10 Marks
	b)	Consider a two-level cache hierarchy L1 and L2 caches. An application incurs 1.4 memory accesses per instruction on average. For this application, the miss rate of L1 cache 0.1, the L2 cache experience on average. 7 misses per 1000 instructions. Calculate the miss rate of L2 expressed correct to two decimal places.	10 Marks

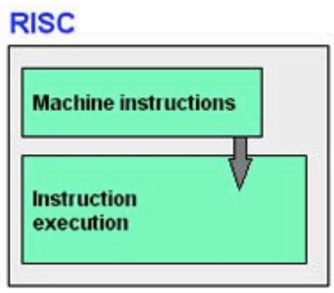
4.	a)	Explain the cache optimization technique for avoiding Address Translation during Indexing of the Cache to Reduce Hit Time.	10 Marks
	b)	State and explain Amdahl's law. Derive an expression for CPU clock as a function of instruction count clocks per instruction and clock cycle time.	10 Marks

5.	a)	Explain the following issues in instruction level parallelism with suitable examples: i) Data dependences ii) Name dependences iii) Control dependences	10 Marks
	b)	List Flynn's taxonomy based multi-processor computer architecture classification. Explain briefly Single Instruction, Single Data (SISD) classification.	10 Marks

SRN

1 a) Differentiate between CISC and RISC type of microprocessors.

Key difference: The main difference between RISC and CISC is in the number of computing cycles each of their instructions take. The difference the number of cycles is based on the complexity and the goal of their instructions.



The term RISC stands for 'Reduced Instruction Set Computer'. It is a CPU design strategy based on simple instructions and fast performance.

RISC is small or reduced set of instructions. Here, each instruction is meant to achieve very small tasks. In a RISC machine, the instruction sets are simple and basic, which help in composing more complex instructions. Each instruction is of the same length; the instructions are strung together to get complex tasks done in a single operation. Most instructions are completed in one machine cycle. This pipelining is a key technique used to speed up RISC machines.

RISC is a microprocessor that is designed to carry out few instructions at the same time. Based on small instructions, these chips require fewer transistors, which make the transistors cheaper to design and produce. Some other features of RISC include

less decoding demand

less decoding demand

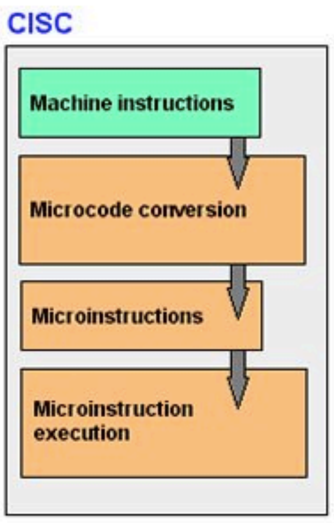
- Uniform instruction set
- Identical general purpose register
- Simple addressing nodes
- Few data types in hardware

Also, while writing codes, RISC makes it easier by allowing the programmer to remove unnecessary codes and prevents wasting of cycles.

The term CISC stands for 'Complex Instruction Set Computer'. It is a CPU design strategy based on single instructions, which are capable of performing multi-step operations.

CISC computers have shorted programs. It has a large number of complex instructions, which takes long time to execute. Here, a single set of instruction is covered in multiple steps; each instruction set has more than three hundred separate instructions. Most instructions are completed in two to ten machine cycles. In CISC, instruction pipelining is not easily implemented.

The CISC machines have good performances, based on the simplification of program compilers; as the range of advanced instructions are easily available in one instruction set. They design complex instructions in one simple set of instructions.



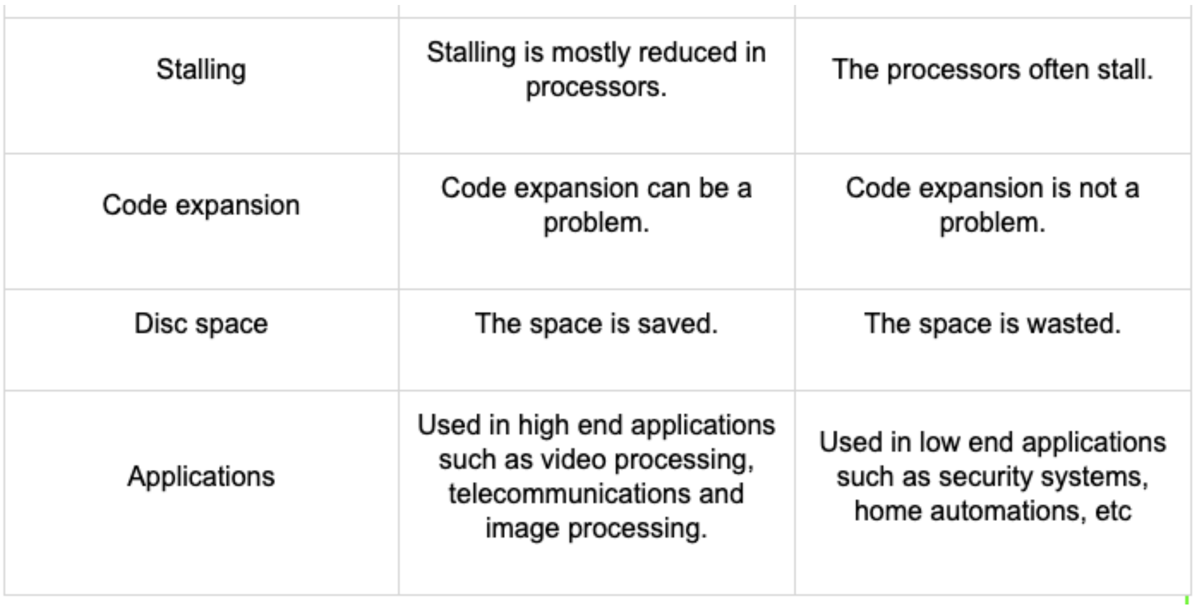
SRN

They perform low level operations such as an arithmetic operation, or a load from memory and memory store. CISC makes it easier to have large addressing nodes and more data types in the machine hardware. However, CISC is considered less efficient than RISC, because of it inefficiency to remove codes which leads to wasting of cycles. Also, microprocessor chips are difficult to understand and program for, because of the complexity of the hardware.

Comparison between RISC and CISC:

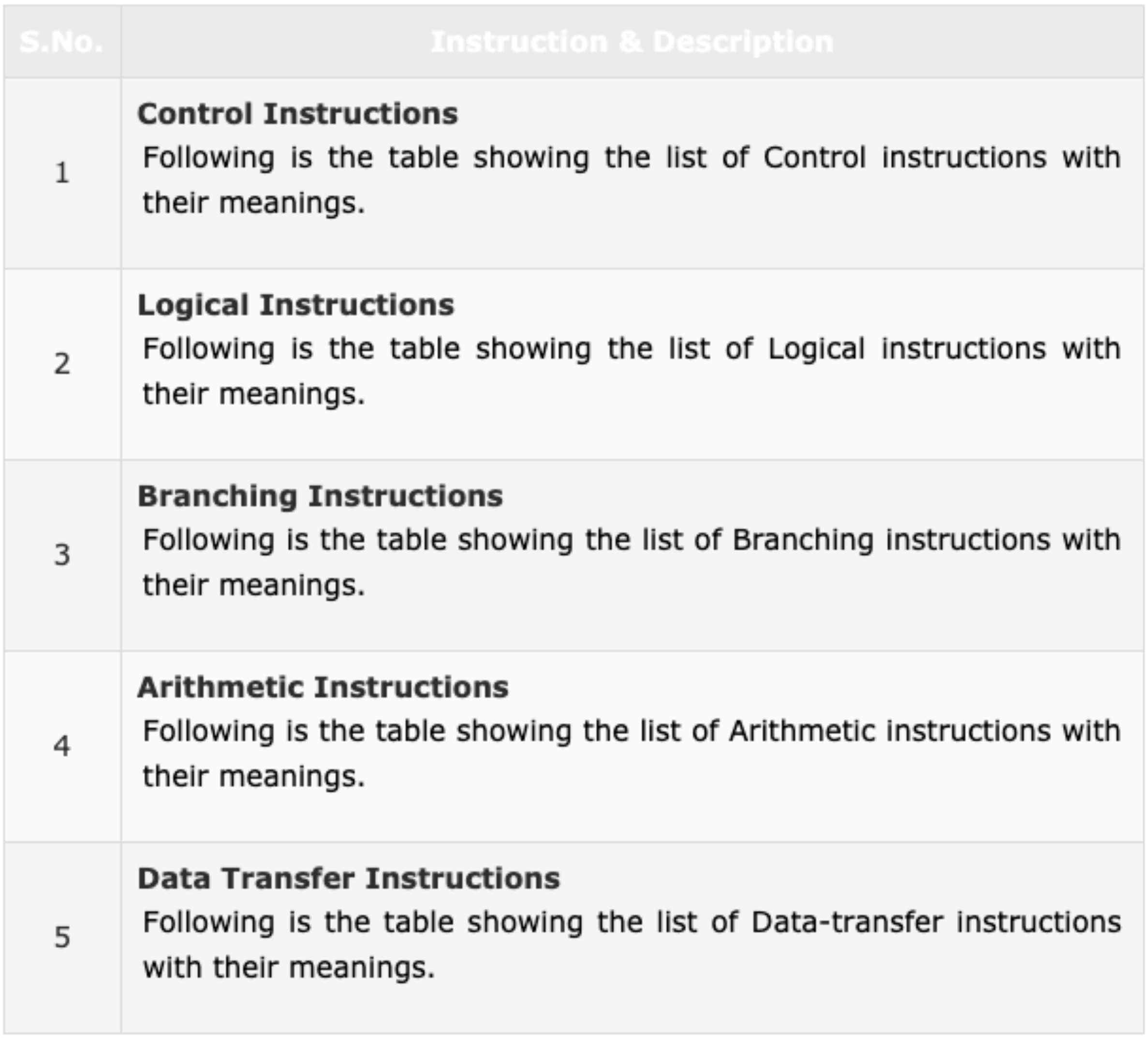
	RISC	**CISC**
Acronym	It stands for 'Reduced Instruction Set Computer'.	It stands for 'Complex Instruction Set Computer'.
Definition	The RISC processors have a smaller set of instructions with few addressing nodes.	The CISC processors have a larger set of instructions with many addressing nodes.
Memory unit	It has no memory unit and uses a separate hardware to implement instructions.	It has a memory unit to implement complex instructions.
Program	It has a hard-wired unit of programming.	It has a micro-programming unit.
Design	It is a complex complier design.	It is an easy complier design.
Calculations	The calculations are faster and precise.	The calculations are slow and precise.
Decoding	Decoding of instructions is simple.	Decoding of instructions is complex.
Time	Execution time is very less.	Execution time is very high.
External memory	It does not require external memory for calculations.	It requires external memory for calculations.
Pipelining	Pipelining does function correctly.	Pipelining does not function correctly.

SRN



Stalling	Stalling is mostly reduced in processors.	The processors often stall.
Code expansion	Code expansion can be a problem.	Code expansion is not a problem.
Disc space	The space is saved.	The space is wasted.
Applications	Used in high end applications such as video processing, telecommunications and image processing.	Used in low end applications such as security systems, home automations, etc

1 b). Explain briefly RISC Instruction Set-Operations.



S.No.	Instruction & Description
1	**Control Instructions** Following is the table showing the list of Control instructions with their meanings.
2	**Logical Instructions** Following is the table showing the list of Logical instructions with their meanings.
3	**Branching Instructions** Following is the table showing the list of Branching instructions with their meanings.
4	**Arithmetic Instructions** Following is the table showing the list of Arithmetic instructions with their meanings.
5	**Data Transfer Instructions** Following is the table showing the list of Data-transfer instructions with their meanings.

SRN

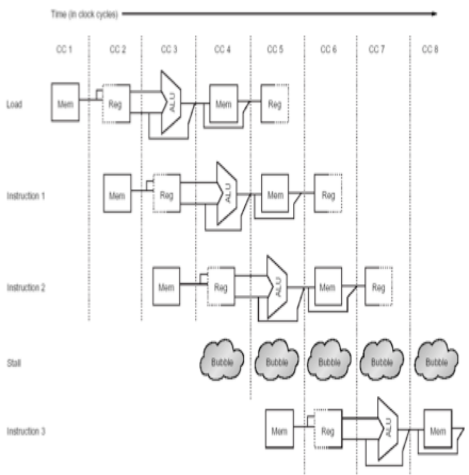
2 a). What are the major hurdles of pipelining? Explain any one of the hazards in detail.

Hurdles that the PIPELINE introduces

- Major functional units are used in different cycles. Hence, overlapping the execution of multiple instructions introduces relatively few conflicts.
  - Separate data and instruction memory – HARVARD architecture.
  - Register file / Register bank is used in two stages – One for reading in ID & writing in WB.
  - PC updating during every clock cycle i.e., Increment and store in PC .
  - Requires adder to compute the potential branch target during the ID stage.
  - Branch does not change PC until the ID stage.
  - Thus creates problem. This will be handled shortly in hazards.
    - Structural , Data & Control Hazards.
  - Ensure that the instructions in the pipeline do not interfere with one another.

Structural Hazards

- If certain combination of instructions can't be accommodated because of resource conflicts, the machine is said to have a *structural hazard.*
- It can be generated by:
  - Some functional unit is not fully pipelined.
  - Some resources has not been duplicated enough to allow all the combinations in the pipeline to execute.
  - Ex: A machine may have only one register file write port, but under certain conditions, the pipeline might want to perform two writes in one clock cycle – this will generate structural hazard.
    - When a sequence of instructions encounter this hazard, the pipeline will stall one of the instructions until the required unit is available.
    - Such stalls will increase the Clock cycle Per Instruction from its ideal 1 for pipelined machines.

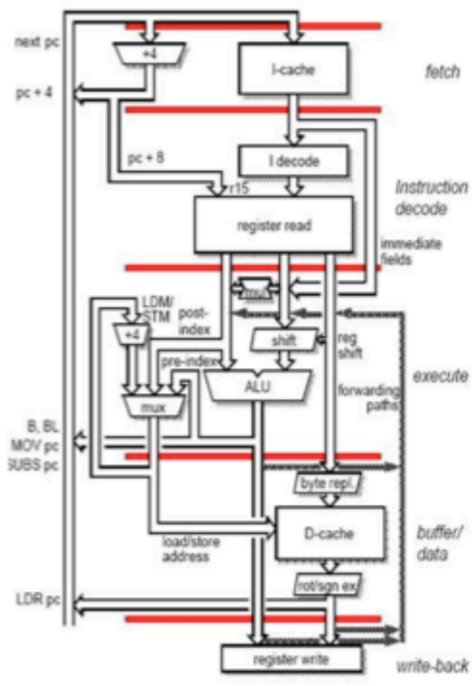


- Stall cycle added (commonly called pipeline *bubble*)

- A machine without structural hazard will have lower CPI
- Why a designer allows structural hazard?
  - To reduce cost
    - Pipelining all the functional units or duplicating them may be too costly
  - To reduce latency
    - Introducing too many pipeline stages may cause latency issues

SRN | | | | | | | | | | | |

2 b). With data path explain classic five-stage pipeline for a RISC processor



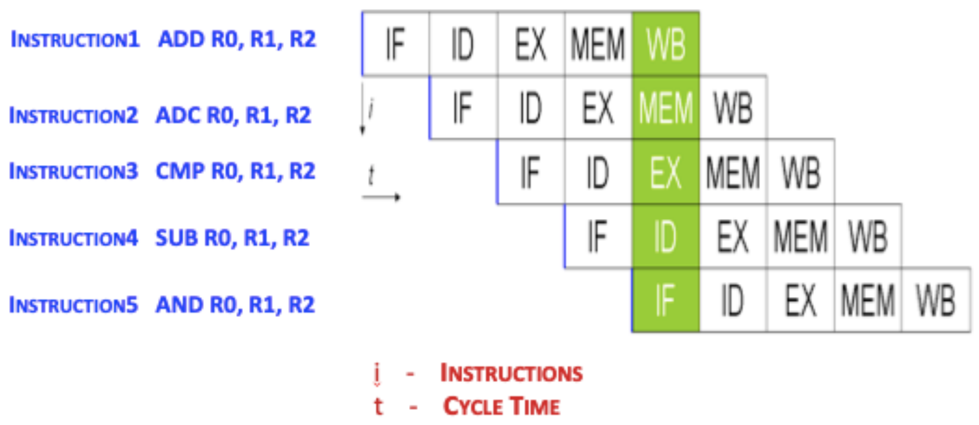
- **Fetch - [IF]**
  - The instruction is **fetched** from memory and placed in the instruction pipeline.
  - Update the PC to the next sequential PC by adding 4 to PC.
- **Decode – [ID]**
  - The instruction is **decoded** and **register operands read** from the register files. There are 3 operand read ports in the register file so most ARM instructions can source all their operands in one cycle.
  - Do the equality test on the registers as they are read, for a possible branch.
  - Sign extend the offset field of the instruction in case it is needed.
  - Compute the possible branch target address by adding the sign-extended offset to the incremented PC.
  - Further, the branch can be completed by the end this stage by storing the branch target address into the PC if condition yielded true.
  - Decoding is done in parallel with reading registers, as the register specifiers are at fixed location in a RISC architecture.
    - Known as fixed –field decoding.
    - Registers may be read which may not be used, that doesn't hurt the performance.

- Also called as execute / effective address cycle.
- The ALU operates on the operands prepared in the previous cycle, performing of the three functions depending on the instruction type.
  - Memory Reference: the ALU adds the base register and the offset to form the effective address.
  - Register – Register ALU instruction: The ALU performs the operation specified by the opcode on the values read from the register file.
  - Register – Immediate ALU instruction: The ALU performs the operations specified by the opcode on the first value read from the register file and the sign extended immediate.

- **Buffer/Data or Memory Access-[MEM]**
  - **Data memory is accessed** if required. Otherwise the ALU result is simply buffered for one cycle.
  - If the instruction is a LOAD, the memory does a read using effective address computed in the previous cycle.
  - If the instruction is a STORE, then the memory writes the data from the second register read using the effective address.
- **Write back – [WB]**
  - The result generated by the instruction are **written back to the register file**, including any data loaded from memory

- **Write back – [WB]**
  - The result generated by the instruction is **written back to the register file**.
  - The data may come either from memory system [for LOAD], or from the ALU [for an ALU instruction].

SRN | | | | | | | | | | | | |


INSTRUCTION1 ADD R0, R1, R2
INSTRUCTION2 ADC R0, R1, R2
INSTRUCTION3 CMP R0, R1, R2
INSTRUCTION4 SUB R0, R1, R2
INSTRUCTION5 AND R0, R1, R2
IF ID EX MEM WB
IF ID EX MEM WB
IF ID EX MEM WB
IF ID EX MEM WB
IF ID EX MEM WB
i
t
i - INSTRUCTIONS
t - CYCLE TIME

3 a). Briefly explain six basic cache optimization schemes.

Cache Optimizations
Six basic cache optimizations
1. Larger block size to reduce miss rate:

- To reduce miss rate through spatial locality.- Increase block size.

- Larger block size reduce compulsory misses.
- But they increase the miss penalty.

2. Bigger caches to reduce miss rate:
- capacity misses can be reduced by increasing the cache capacity.
- Increases larger hit time for larger cache memory and higher cost and power.

3. Higher associativity to reduce miss rate:
- Increase in associativity reduces conflict misses.

4. Multilevel caches to reduce penalty:
- Introduces additional level cache
- Between original cache and memory.
- L1- original cache

L2- added cache.
L1 cache: - small enough
- speed matches with clock cycle time.

L2 cache: - large enough
- capture many access that would go to main memory.

Average access time can be redefined as
Hit timeL1+ Miss rate L1 X (Hit time L2 + Miss rate L2 X Miss penalty L2)

5. Giving priority to read misses over writes to reduce miss penalty:
- write buffer is a good place to implement this optimization.
- write buffer creates hazards: read after write hazard.

6. Avoiding address translation during indexing of the cache to reduce hit time:
- Caches must cope with the translation of a virtual address from the processor to a physical address to access memory.
- common optimization is to use the page offset.
- part that is identical in both virtual and physical addresses- to index the cache.

SRN

3 b). Consider a two-level cache hierarchy L1 and L2 caches. An application incurs 1.4 memory accesses per instruction on average. For this application, the miss rate of L1 cache 0.1, the L2 cache experience on average. 7 misses per 1000 instructions. Calculate the miss rate of L2 expressed correct to two decimal places.

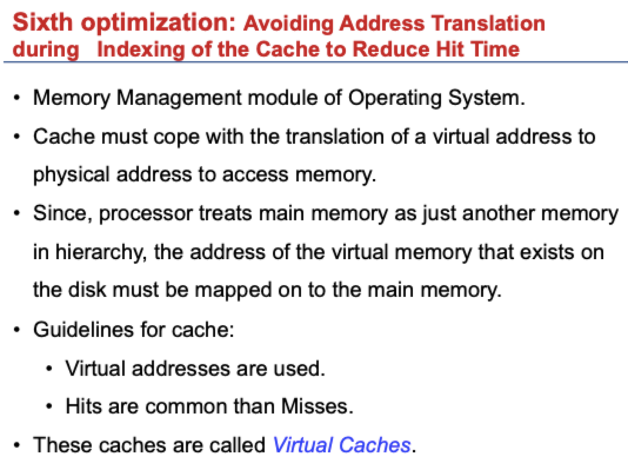
Let 1000 instructions generated 121 Therefore, Miss rate of L2 cache = (memory references generated by L2 cache) / (memory references generated by L1 cache) = 7 / 140 = 0.05 Alternate Solution

On Average, 1.4 memory accesses are required for one instruction execution.
So, for 1000 instructions, 1400 accesses are needed.

Number of misses occurred in cache L2 for 1000 instruction = 7/1400 = 0.005

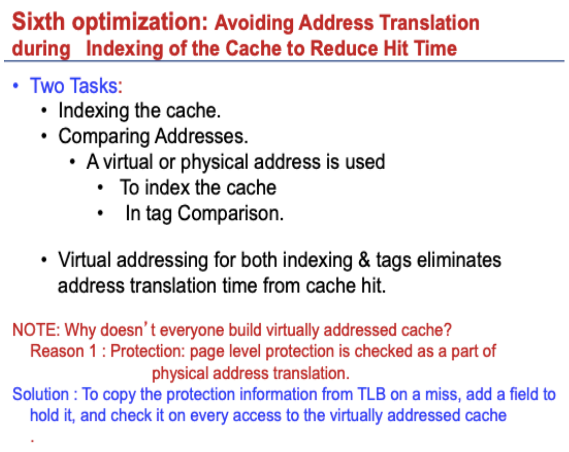
Miss rate of L2 cache = misses occurred in L2 cache / miss rate in L1 cache
= 0.005 / 0.1 = 0.05

4 a) Explain the cache optimization technique for avoiding Address Translation during Indexing of the Cache to Reduce Hit Time.



Sixth optimization: Avoiding Address Translation during Indexing of the Cache to Reduce Hit Time

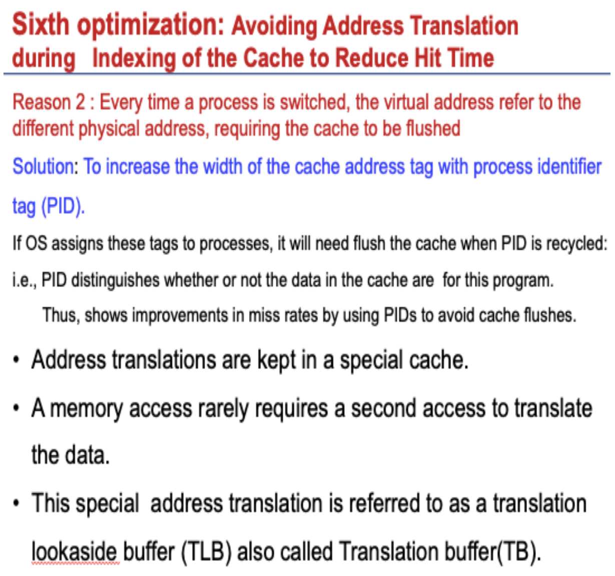
- Memory Management module of Operating System.
- Cache must cope with the translation of a virtual address to physical address to access memory.
- Since, processor treats main memory as just another memory in hierarchy, the address of the virtual memory that exists on the disk must be mapped on to the main memory.
- Guidelines for cache:
  - Virtual addresses are used.
  - Hits are common than Misses.
- These caches are called *Virtual Caches*.



Sixth optimization: Avoiding Address Translation during Indexing of the Cache to Reduce Hit Time

- Two Tasks:
  - Indexing the cache.
  - Comparing Addresses.
    - A virtual or physical address is used
      - To index the cache
      - In tag Comparison.
  - Virtual addressing for both indexing & tags eliminates address translation time from cache hit.

NOTE: Why doesn't everyone build virtually addressed cache?
Reason 1 : Protection: page level protection is checked as a part of physical address translation.
Solution : To copy the protection information from TLB on a miss, add a field to hold it, and check it on every access to the virtually addressed cache



Sixth optimization: Avoiding Address Translation during Indexing of the Cache to Reduce Hit Time

Reason 2 : Every time a process is switched, the virtual address refer to the different physical address, requiring the cache to be flushed

Solution: To increase the width of the cache address tag with process identifier tag (PID).

If OS assigns these tags to processes, it will need flush the cache when PID is recycled: i.e., PID distinguishes whether or not the data in the cache are for this program.
Thus, shows improvements in miss rates by using PIDs to avoid cache flushes.

- Address translations are kept in a special cache.
- A memory access rarely requires a second access to translate the data.
- This special address translation is referred to as a translation lookaside buffer (TLB) also called Translation buffer(TB).

Virtual Address to Physical Address

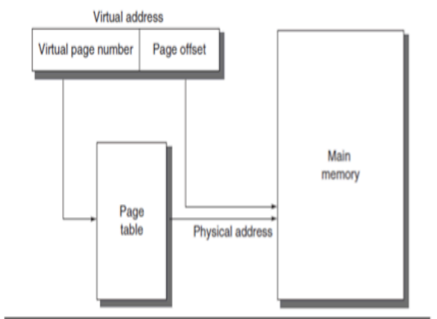


Figure B.23 The mapping of a virtual address to a physical address via a page table.

SRN | | | | | | | | | | | |

4b) State and explain Amdahl's law. Derive an expression for CPU clock as a function of instruction count clocks per instruction and clock cycle time.

In computer architecture, Amdahl's law is a formula which gives the theoretical speedup in latency of the execution of a task at fixed workload that can be expected of a system whose resources are improved. It is named after computer scientist Gene Amdahl, and was presented at the AFIPS Spring Joint Computer Conference in 1967.

Most computers run synchronously utilizing a CPU clock running at a constant clock rate: where: Clock rate = 1 / clock cycle
• The CPU clock rate depends on the specific CPU organization (design) and hardware implementation technology (VLSI) used.
• A computer machine (ISA) instruction is comprised of a number of elementary or micro operations which vary in number and complexity depending on the instruction and the exact CPU organization (Design).

- A micro operation is an elementary hardware operation that can be performed during one CPU clock cycle.
- This corresponds to one microinstruction in micro programmed CPUs.
- Examples: register operations: shift, load, clear, increment, ALU operations: add, subtract, etc.
- Thus a single machine instruction may take one or more CPU cycles to complete termed as the Cycles Per Instruction (CPI).
- Average (or effective) CPI of a program: The average CPI of all instructions executed in the program on a given CPU design.
- Cycle 1 cycle 2 cycle 3 Cycles/sec = Hertz = Hz MHz = 106 Hz GHz = 109 Hz

5 a) Explain the following issues in instruction level parallelism with suitable examples:
i) Data dependences ii) Name dependences iii) Control dependences

Data Dependences
An instruction j is data dependant on instruction i if either of the following holds:
i) Instruction i produce a result that may be used by instruction j
Eg1: i: L.D F0, 0(R1)
j: ADD.D F4, F0, F2

ith instruction is loading the data into the F0 and jth instruction use F0 as one the Operand. Hence, jth instruction is data dependant on ith instruction.
Eg2: DADD R1, R2, R3
DSUB R4, R1, R5

ii) Instruction j is data dependant on instruction k and instruction k data dependant on
Instruction i
Eg: L.D F4, 0(R1)
MUL.D F0, F4, F6
ADD.D F5, F0, F7

SRN

Dependences are the property of the programs. A Data value may flow between Instructions either through registers or through memory locations. Detecting the data flow and dependence that occurs through registers is quite straightforward. Dependences that flow through the memory locations are more difficult to detect. Data dependence convey three things.
a) The possibility of the Hazard.
b) The order in which results must be calculated and
c) An upper bound on how much parallelism can possibly exploited.

Name Dependence occurs when two instructions use the same Register or Memory location, but there is no flow of data between the instructions associated with that name.

Two types of Name dependences:
i) Antidependence: between instruction i and instruction j occurs when instruction j writes a register or memory location that instruction i read. He original ordering must be preserved to ensure that i read the correct value.
Eg: L.D F0, 0(R1)
DADDUI R1, R1, R3

ii) Output dependence: Output Dependence occurs when instructions i and j write to the same register or memory location.
Ex: ADD.D F4, F0, F2
SUB.D F4, F3, F5
The ordering between the instructions must be preserved to ensure that the value finally written corresponds to instruction j.The above instruction can be reordered or can be executed simultaneously if the name of the register is changed. The renaming can be easily done either statically by a compiler or dynamically by the hardware.

Data hazard: Hazards are named by the ordering in the program that must be preserved by the pipeline
RAW (Read After Write): j tries to read a source before i write it, so j in correctly gets old value, this hazard is due to true data dependence.
WAW (Write After Write): j tries to write an operand before it is written by I.
WAW Hazard arises from output dependence.
WAR (Write After Read): j tries to write a destination before it is read by I, so that I Incorrectly gets the new value. WAR hazard arises from interdependence and normally Cannot occur in static issue pipeline.

CONTROL DEPENDENCE:

A control dependence determines the ordering of an instruction i with respect to a branch Instruction,
Ex: if P1 {
S1;
}
if P2 {
S2;
}

SRN | | | | | | | | | | | |

S1 is Control dependent on P1 and
S2 is control dependent on P2 but not on P1.
a) An instruction that is control dependent on a branch cannot be moved before the branch ,so that its execution is no longer controlled by the branch.
b) An instruction that is not control dependent on a branch cannot be moved after the branch so that the branch controls its execution.

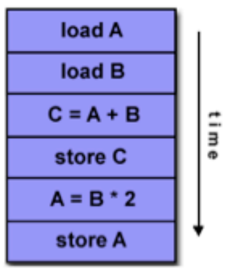
5 b) List Flynn's taxonomy based multi-processor computer architecture classification. Explain briefly Single Instruction, Single Data (SISD) classification.

- The matrix below defines the 4 possible classifications according to Flynn

SISD Single Instruction, Single Data	SIMD Single Instruction, Multiple Data
MISD Multiple Instruction, Single Data	MIMD Multiple Instruction, Multiple Data

Single Instruction, Single Data (SISD)

- A serial (non-parallel) computer
- Single instruction: only one instruction stream is being acted on by the CPU during any one clock cycle
- Single data: only one data stream is being used as input during any one clock cycle
- Deterministic execution
- This is the oldest and until recently, the most prevalent form of computer
- Examples: most PCs, single CPU workstations and mainframes


load A
load B
C = A + B
store C
A = B * 2
store A
time