DMI COLLEGE OF ENGINEERING
DEPARTMENT OF ELCTRONICS AND COMMUNICATION ENGINEERING
CS6303 – COMPUTER ARCHITECTURE UNIVERSITY QUESTION PAPER
UNIT I
PART A

1) State Amdahl's law? **(Nov/Dec 2014)**

Amdahl's law, also known as **Amdahl's** argument, is used to find the maximum expected improvement to an overall system when only part of the system is improved. It is often used in parallel computing to predict the theoretical maximum speed up using multiple processors.

If we take spped up S(n) to be b/bs then we have

$$S(n)=\frac{n}{1+(n-1)F}$$

F- non parallelization operation.

2) Brief about relative addressing mode with an example? **(Nov/Dec 2014)**

It is same as index mode. The difference is, instead of general purpose register, here we can use program counter(PC).

Relative Mode: The Effective Address is determined by the Index mode using the PC in place of the general purpose register (gpr). This mode can be used to access the data operand. But its most common use is to specify the target address in branch instruction.

Eg. Branch>0 Loop It causes the program execution to goto the branch target location. It is identified by the name loop if the branch condition is satisfied.

Syntax of Relative Addressing mode is: X(PC) EA=[PC]+X

3) List the eight great ideas invented by computer architects? **(Apr/May 2015)**

a) Design for Moore's Law
b) Use abstraction to simplify design
c) Make the common case first
d) Performance via Parallelism
e) Performance via Pipelining
f) Performance via Prediction
g) Hierarchy of Memory
h) Dependability via Redundancy
i)

4) Distinguish pipelining from parallelism? **(Apr/May 2015)**

In computing, a pipeline is a set of data processing elements connected in series, where the output of one element is the input of other. Pipelining is a form of parallelism.

Parallelism is simply, multiple operations happening at same time

5) What are R-Type instructions? **(Apr/may 2015)**

R instructions are used when all the data values used by the instruction are located in registers.

The syntax of the R-Type is given below

Op	Rs	Rt	Rd	Shamt	function
6	5	5	5	5	6

Where "OP" is the mnemonic for the particular instruction. *rs*, and *rt* are the source registers, and *rd* is the destination register

6) What is instruction set architecture? **(Nov/Dec 2015)**

The ***Instruction Set Architecture*** (ISA) is the part of the processor that is visible to the programmer or compiler writer. The ISA serves as the boundary between software and hardware. The 3 most common types of ISAs are:

1. ***Stack*** - The operands are implicitly on top of the stack.
2. ***Accumulator*** - One operand is implicitly the accumulator.
3. *General Purpose Register (GPR)* - All operands are explicitely mentioned, they are either registers or memory locations
4.

7) How CPU execution time for a program in calculated? **(Nov/Dec 2015)**

CPU execution time for a program = CPU clock cycle for a program X Clock cycle time

8) How to represent instruction in a computer system? **(May/Jun 2016)**

An instruction is an order given to a computer processor by a computer program. At the lowest level, each instruction is a sequence of 0s and 1s that describes a physical operation the computer is to perform (such as "Add") and, depending on the particular instruction type, the specification of special storage areas called registers that may contain data to be used in carrying out the instruction, or the location in computer memory of data.

9) Distinguish between auto increment and auto decrement addressing mode? **(May/Jun 2016)**

The Effective Address of the operand is the contents of a register in the instruction. After accessing the operand, the contents of this register is automatically incremented.

The Effective Address of the operand is the contents of a register in the instruction. After accessing the operand, the contents of this register is automatically decremented to point to the next item in the list.

10) What are the components or the functional units of a computer system?

a) Input Unit
b) Output Unit
c) Memory Unit
d) Central processing Unit
   i. Arithmetic and logical Unit
   ii. Control Unit
   iii.

11) Define different types of Data Transfer Instructions?

A command that moves data between memory and registers.

Load: The data transfer instructions that copies data from memory to a register is traditionally called load.The format of the load instruction is the name of the operation followed by the register to be loaded,then a constant and register used to memory access.

Store: The data transfer instructions that copies data from register to memory is traditionally called store.The format of thestore instruction is the name of the operation followed by the register to be stored,then offset to select the array element and finally the base register.

12) Write an example for immediate operand.

The quick add instruction with one constant operand is called add immediate or addi.To add 4 to register $s3, we just writeaddi $s3,$s3,4 # $s3 = $s3 + 4.

13) What is Conditional branch?

An instruction that requires the comparison of two values and that allows for a subsequent transfer of control to new address in the program based on the outcome of the comparison.

14) Define Addressing Modes?

The different ways in which the operands of an instruction are specified are called as addressing modes. The MIPS addressing modes are the following:

1. Immediate addressing
2. Register addressing
3. Base or displacement addressing
4. PC-relative addressing
5. Pseudo direct addressing

15) Define Registers?

In computer architecture, a processor register is a small amount of storage available on the CPU whose contents can be accessed more quickly than storage available elsewhere. Modern computer architectures operate on the principle of moving data from main memory into registers, operating on them, then moving the result back into main memory. Registers are at the top of the memory hierarchy, and provide the fastest way for a CPU to access data.

16) Define ISA?

The instruction set architecture, or simply architecture of a computer is the interface between the hardware and the lowest-level software. It includes anything programmers need to know to make a binary machine language program workcorrectly, including instructions, I/O devices, and so on.

17) Define– Stored Program Concepts

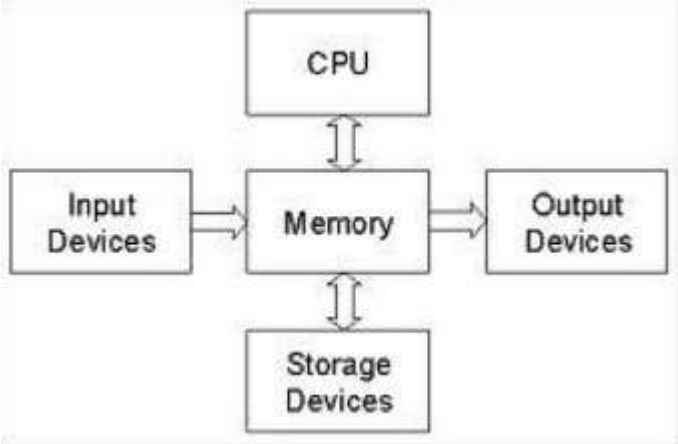
Today's computers are built on two key principles:

- Instructions are represented as numbers.
- Programs are stored in memory to be read or written, just like data.

These principles lead to the stored-program concept. Treating instructions in the same way as data greatly simplifies both the memory hardware and the software of computer systems.

PART B

1. Explain the components of computer system? **(Nov/Dec 2014) (Nov/Dec 2015)**



Processor (CPU), Main memory, Secondary memory, Input devices, Output devices, bus, Software, Application programs, System Programs.

2. State the CPU performance equation and discuss the factors that affect performance? **(Nov/Dec 2014)**

Response time (aka execution time) = the time between the start and the completion of a task
$performance_X = 1 / execution\_time_X$

Speedup

$$\frac{performanceX}{performanceY} = \frac{execution\_timeY}{execution\_timeX} = n$$

Throughput – the total amount of work done in a given time

CPU execution time (CPU time) – time the CPU spends working on a task

CPU execution time for a program=CPU Clock cycles for a program x clock cycle time

$$\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock Rate}}$$

CPU Clock cycles for a program=#Instructions for a program x Average clock cycles per instruction

$$\text{CPU execution time for a program} = \frac{performanX}{performanceY} = \frac{execution\_timeY}{execution\_timeX} = n$$

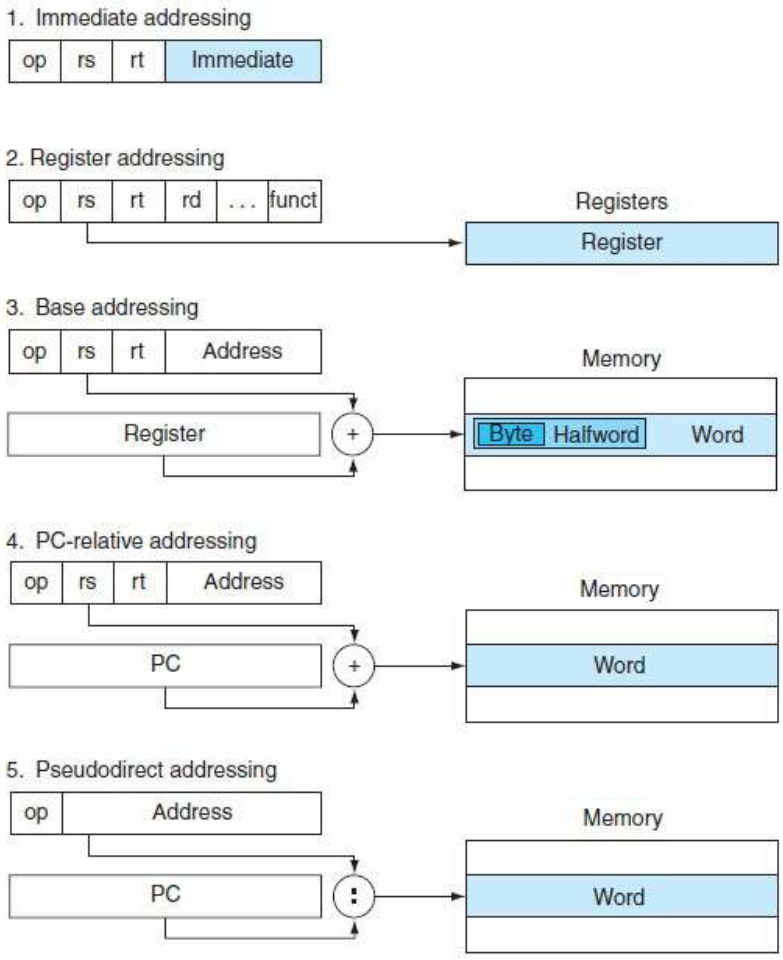
$$\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock Rate}}$$

$$\text{Overall effective CPI} = \sum_{i=1}^{n} (CPI_i \times IC_i)$$

CPU time = Instruction_count x CPI x clock_cycle

$$\text{CPU time} = \frac{\text{Instuction count x CPI}}{\text{Clock Rate}}$$

3. What is the need for addressing in a computer system? Explain the different addressing modes with suitable examples? **(Apr/May 2015)(Nov/Dec 2015)**



Discuss about the various techniques to represent instructions in a computer system?

(Apr/May 2015)

4. Assume a two address format specified as source, destination. Examine the following sequence of instruction and explain the addressing modes used and the operation done in every instruction?

(Nov/Dec 2014)

1. Move (R5) +, R0
2. Add (R5) +, R0.
3. Move (R0), (R5)
4. Move 16(R5), R3
5. Add #40, R5

Answer:

1. ***Move (R5)+, R0***;

When + comes after register addressing then it has to be auto incremented after the action

So move the value from the address pointed by R5 to R0 and then the increment R5 to next address

2. ***Add (R5)+, R0***

Add the value stored at address pointed by R5 with R0 and then increment R5 address value

3. ***Move (Ro), (R5)***

Move the value from the address R0 to the address pointed in R5

4. ***Move 16(R5), R3*** – Indexed mode

Move the value from the address pointed by 16 bits offset of R5 register

5. ***Add #40, R5***

It will store the value 40 directly to register R5

6. Consider the computer with three instruction classes and CPI measurement as given below and Instruction counts for each instruction class for the same program from two different compilers are given. Assume that the computer's clock rate is 4Ghz. Which code sequence will execute faster according to execution time? **(Nov/Dec 2014)**

Code from	CPI for this Instruction class		
	A	B	C
CPI	1	2	3
Code from	Instruction count for each class		
	A	B	C
Compiler 1	2	1	2
Compiler 2	4	1	1

Answer

i. Instructions:

Instruction executed by Compiler 1 = 2+1+2 = 5

Instruction executed by Compiler 2 = 4+1+1 = 6

ii. CPU clock cycles:

CPU clock cycle = IC X CPI

CPU clock cycle for Compiler 1 = (2 X 1) + (1 X 2) + (2 X 3)

= 2 + 2 + 6

= 10 cycles

CPU clock cycle for Compiler 2 = (4 X 1) + (1 X 2) + (1 X 3)

= 4 + 2 + 3

= 9 cycles

iii. Time for executing the instruction

CPU Execution time = $\frac{CPU\ clock\ cycle}{Clock\ Rate}$

Given: Clock rate = 4GHz = 4 X 10^9 Hz

CPU Execution time for Compiler 1 = $\frac{10}{4 X 10^9}$ = 2.5 X 10^{-9} second

CPU Execution time for Compiler 2 = $\frac{9}{4 X 10^9}$ = 2.25 X 10^{-9} second

Execution time for Compiler 1 > Execution time for Compiler 2

ie, Execution time of Compiler 2 is lesser than Compiler 1,

So Compiler 2 is faster than Compiler 1

UNIT II

1. Define ALU? **(May/Jun 2016)**

   An **arithmetic logic unit** (**ALU**) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer.

2. Define Little Endian arrangements? **(Nov/Dec2014)**

   Little-endian describes the order in which a sequence of bytes is stored in computer memory. Little endian is an order in which the "little end" (least significant value in the sequence) is stored first.

   For example, in a little-endian computer, the two bytes required for the hexadecimal number 4F52 would be stored as 524F (52 at address 1000, 4F at 1001).

3. What are the overflow/ underflow conditions for addition and subtraction?

   (Nov/Dec 2015)

Operation	Operand A	Operand B	Result indicating overflow
A+B	>=0	>=0	<0
A+B	<0	<0	>=0
A-B	>=0	<0	<0
A-B	<0	>=0	>=0

OR

Overflow rule for addition

(+A) + (+B) = -C

(-A) + (-B) = +C

Overflow rule for subtraction

(+A) – (-B) = -C

(-A) – (+B) = +C

4. How overflow occurs in subtraction? **(Apr/May 2015)**

   In a computer, the condition that occurs when a calculation produces a result that is greater in magnitude than which a given register or storage location can store or represent.

5. What do you means by sub word parallelism? **(Apr/May 2015)**

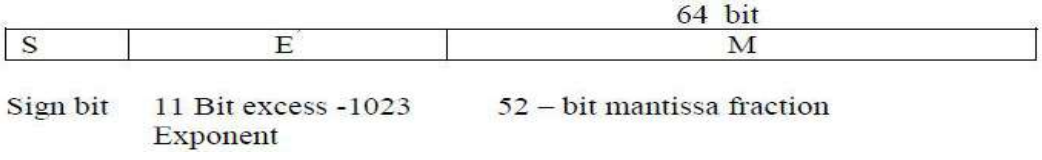
   Sub word parallelism is a technique that enables the full use of word-oriented data paths when dealing with lower precision data. It is a form of low-cost, small-scale SIMD parallelism.

5. What is DMA? **(Nov/Dec 2014)**

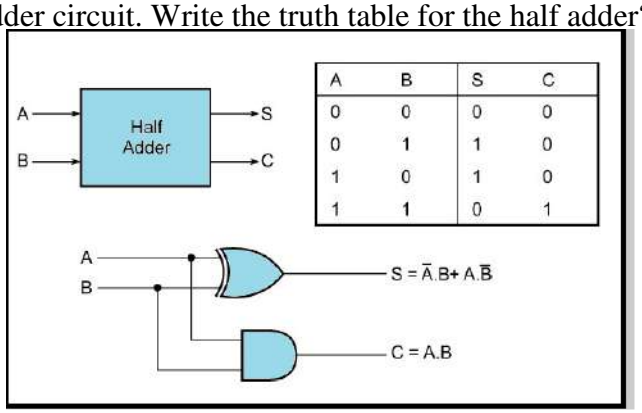
   Direct Memory Access is a technique for transferring data from main memory to a device without passing it through the CPU

7. State the representation of double precision floating point number? **(Nov/Dec 2015)**

   Double precision representation contains 11 bits excess -1023 exponent E' which has the range $1 \leq E' \leq 2046$ for normal values. This is the actual exponent E is in range $-1022 \leq E \leq 1023$. The 53 bit mantissa provides a precision equivalent about 16 decimal digits.



8. Draw the half adder circuit. Write the truth table for the half adder?



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

9. Write down the advantages of Booth's Algorithm
   i. It handles both positive and negative numbers
   ii. It achieves some efficiency in the number of additions required when the multiplier has a few larger blocker of 1's

10) What is a carry look-ahead adder?

The input carry needed by a stage is directly computed from carry signals obtained from all the preceding stages i-1,i-2,…..0, rather than waiting for normal carries to supply slowly from stage to stage. An adder that uses this principle is a called carry look-ahead adder.

11) What are the steps in the floating-point addition?

The steps in the floating-point addition are:

1. Align the decimal point of the number that has the smaller exponent.
2. Addition of the significands.
3. Normalize the sum.
4. Round the result

12) How can we speed up the multiplication process?

There are two techniques to speed up the multiplication process:

1) The first technique guarantees that the maximum number of summands that must be added is n/2 for n-bit operands.

2) The second technique reduces the time needed to add the summands.

13) Write the algorithm for restoring division?

Do the following for n times:

1) Shift A and Q left one binary position.

2) Subtract M and A and place the answer back in A.

3) If the sign of A is 1, set q0 to 0 and add M back to A.

Where A- Accumulator, M- Divisor, Q- Dividend.

14) Define carry save addition (CSA) process.

Instead of letting the carries ripple along the rows, they can be saved and introduced into the next roe at the correct weighted position. Delay in CSA is less than delay through the ripple carry adder.

15) Define Overflow & Underflow?

A situation in which a positive exponent becomes too large to fit in the exponent field.

A situation in which a negative exponent becomes too large to fit in the exponent field.

16) Define Exception or Interrupt?

An unscheduled event that disrupts program execution. The address of the instruction that overflowed is saved in a register, and the computer jumps to a predefined address to invoke the appropriate routine for that exception.

17) What are the floating point instructions in MIPS?

MIPS supports the IEEE 754 single precision and double precision formats with these instructions:

- Floating-point addition
- Floating-point subtraction
- Floating-point multiplication
- Floating-point division
- Floating-point comparison
- Floating-point branch

18) What is meant by sub-word parallelism?

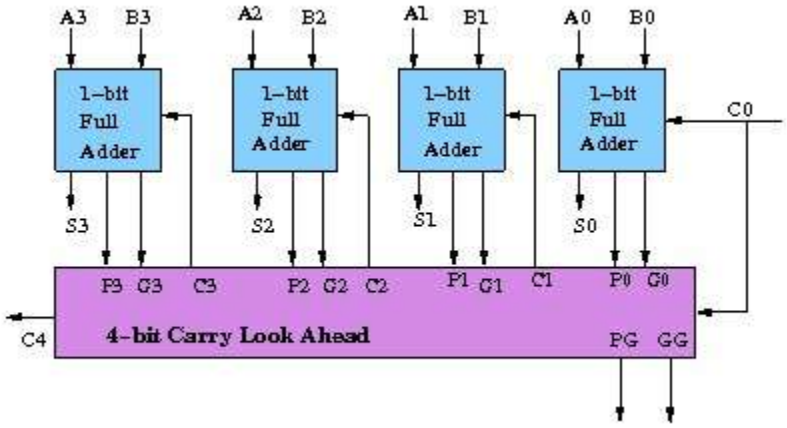
Given that the parallelism occurs within a wide word, the extensions are classified as sub-word parallelism. It is also classified under the more general name of data level parallelism.
They have been also called vector or SIMD, for single instruction, multiple data . The rising popularity of multimedia applications led to arithmetic instructions that support narrower operations that can easily operate in parallel.

PART B

1. Briefly Explain carry look ahead adder? **(Nov/Dec 2014)**

   To reduce the computation time, there are faster ways to add two binary numbers by using carry lookahead adders. They work by creating two signals P and G known to be Carry Propagator and Carry Generator.

   The **carry propagator** is propagated to the next level whereas the **carry generator** is used to generate the output carry , regardless of input carry.



In the carry-lookahead circuit we ned to generate the two signals carry propagator(P) and carry generator(G),

$P_i = A_i \oplus B_i$

$G_i = A_i \cdot B_i$

The output sum and carry can be expressed as

$Sum_i = P_i \oplus C_i$

$C_i+1 = G_i + (P_i \cdot C_i)$

Having these we could design the circuit. We can now write the Boolean function for the carry output of each stage and substitute for each C_i its value from the previous equations:

$C_1 = G_0 + P_0 \cdot C_0$
$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$
$C_3 = G_2 + P_2 \cdot C_2 = G_2\ P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$
$C_4 = G_3 + P_3 \cdot C_3 = G_3\ P_3 \cdot G_2\ P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$

2. Explain the sequential version of multiplication algorithm and it hardware?

(Apr/May 2015)

Steps:

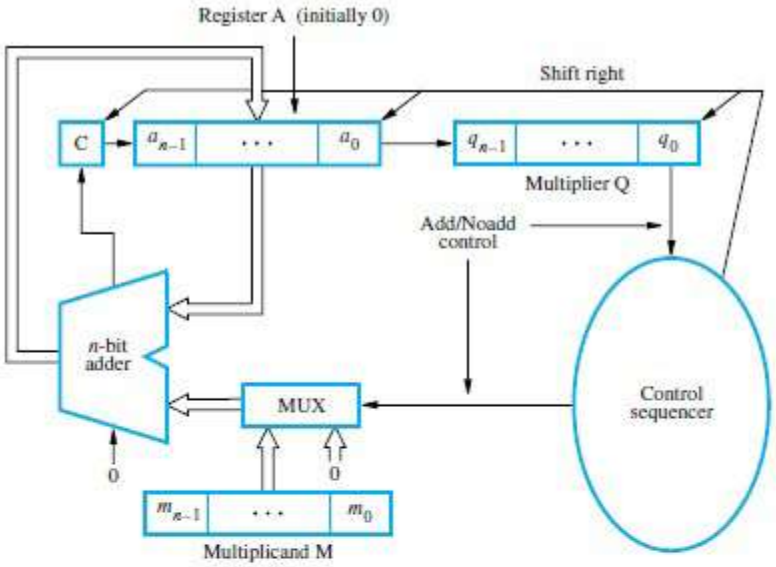
Set B= Multiplicand, Q= Multiplier, A=0, N=Count

If Q_0=1 then A=A+B

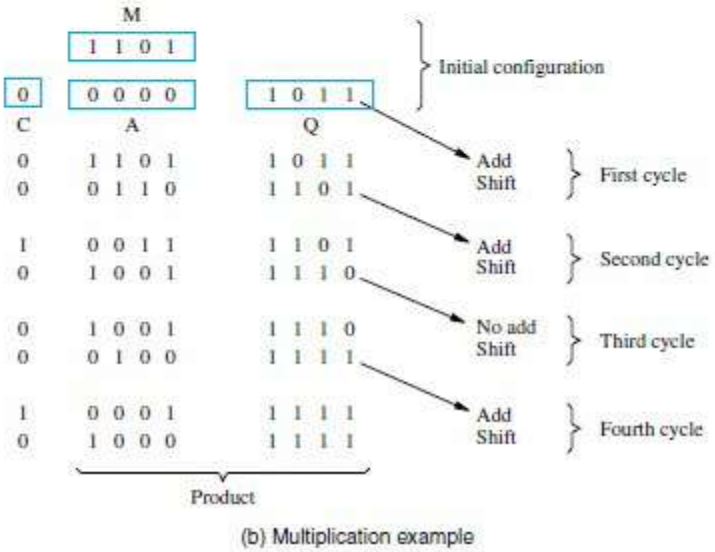
Shift A,Q right

N=N-1 (until N=0)

A,Q is the product.



(a) Register configuration



(b) Multiplication example

3. Discuss in detail about division algorithm in detail with diagram and example?

(Nov/Dec 2015)

Restoring Division

Steps:

1. Set A=0, B=Divisor, Q=Dividend, Count=n;
2. Shift left A,Q;
3. A=A-B;
4. If A<0?
    a. If NO
        i. Q_0=1;
    b. If YES
        i. Q_0=0;
        ii. A=A+B;
5. Count = Count -1
6. If Count ≠0
    a. Go to Step 2
7. Q = Quotient
   A=Remainder

```
      10
11 ) 1000
     11
     --
     10
```

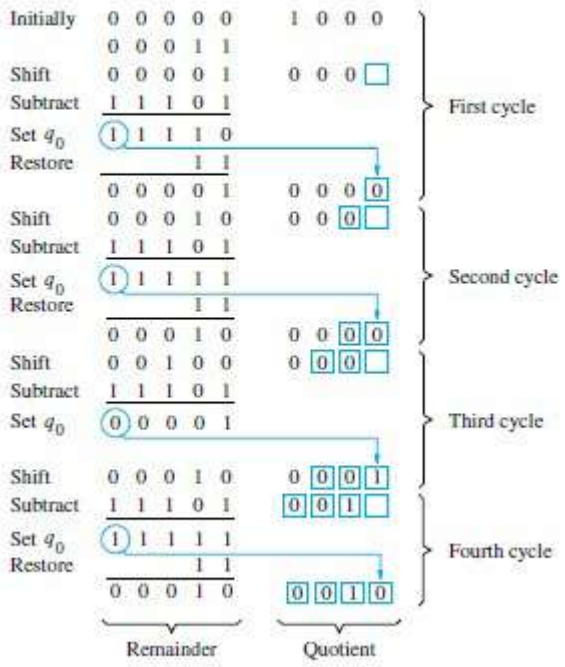


Figure 9.24 A restoring division example.

Non- Restoring Division

Steps:

1. Set A=0, B=Divisor, Q=Dividend, Count=n;
2. Shift left A,Q
3. If A<0?
    a. If YES
        i. A=A+B
    b. If NO
        i. A=A-B
4. If A<0?

a. If YES
    i. Q_0=0;
b. If No
    i. Q_0=1;

5. Count=Count-1
6. If Count =0
    a. If NO
        i. Go To Step 2
    b. If YES
        i. If A<0?
            1. If YES
                a. A=A+B
            2. If NO
                a. END
7. Q=Quotient
   A=Remainder

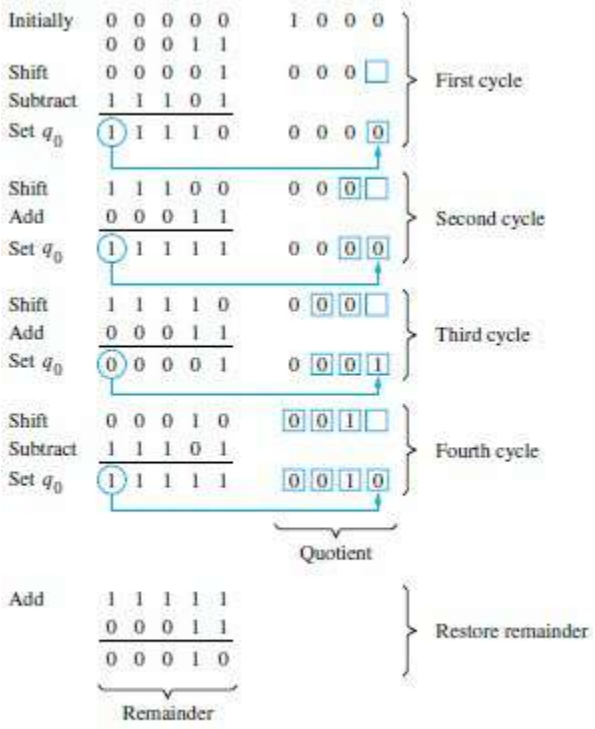


Figure 9.25 A non-restoring division example.

4. Explain briefly about the floating point addition and subtraction in detail?

(May/Jun 2016)

Examples:

$X = 0.3 \times 10^2 = 30$
$Y = 0.2 \times 10^3 = 200$

$X + Y = (0.3 \times 10^{2-3} + 0.2) \times 10^3 = 0.23 \times 10^3 = 230$
$X - Y = (0.3 \times 10^{2-3} - 0.2) \times 10^3 = (-0.17) \times 10^3 = -170$
$X \times Y = (0.3 \times 0.2) \times 10^{2+3} = 0.06 \times 10^5 = 6000$
$X \div Y = (0.3 \div 0.2) \times 10^{2-3} = 1.5 \times 10^{-1} = 0.15$

5. Multiply the following pair of signed nos. using Booth's bit – pair recoding of the multiplier. A = +13(Multiplicand) and B = -6 (Multiplier)? **(Nov/Dec 2014)**

Answer:

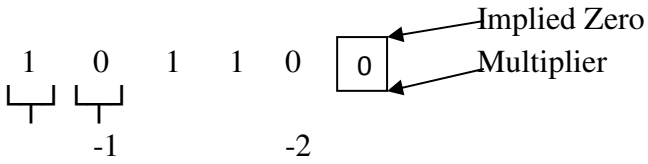
$+ 13_{10}$ = 0 1 1 0 1_2 ← Multiplicand

$- 6_{10}$ = 1 1 1 0_2 ← Multiplicand

↓

1 0 1 0 ← 2's Complement

Recode the multiplier using bit pair recoding method



				0	1	1	0	1
						-1		-2
	1	1	1	0	0	1	1	0
	1	1	0	0	1	1		
1	1	0	1	1	0	0	1	0

6. Divide $(12)_{10}$ by $(3)_{10}$ using the restoring and Non-restoring division algorithm with step by step intermediate results and explain? **(Nov/Dec 2015)**

QUESTION BANK – UNIT III

2 MARKS

1. What is the need for speculation?

Speculation is an approach that allows the compiler or the processor to "guess" about the properties of an instruction, so as to enable execution to begin for other instructions that may depend on the speculated instruction. For example, we might speculate on the outcome of a branch, so that instructions after the branch could be executed earlier. Or, we might speculate that a store that precedes a load does not refer to the same address, which would allow the load to be executed before the store. The difficulty with speculation is that it may be wrong

2. What is exception?

An exception is an unexpected event from within the processor; arithmetic overflow is an example of an exception. The basic action that the machine must perform when an exception occurs is to save the address of the offending instruction in the exception program counter (EPC) and then transfer control to the operating system at some specified address.

3. What is a branch prediction?

Branch prediction is a method of resolving a branch hazard that assumes a given outcome for the branch and proceeds from that assumption rather than waiting to ascertain the actual outcome.

4. What is meant by branch prediction buffer?

It is a small memory indexed by the lower portion of the address of the branch instruction. The memory contains a bit that says whether the branch was recently taken or not.

5. What are the advantages of pipelining?

More instruction can be executed in same time

Reduce the time needed for execution

Efficient usage of the processors

6. Define pipelining.

Pipelining is a technique of decomposing a sequential process into sub operations with each sub process being executed in a special dedicated segment that operates concurrently with all other segments.

7. What are the steps required for a pipelined processor to process the instruction?

- F Fetch: read the instruction from the memory
- D Decode: decode the instruction and fetch the source operand(s).
- E Execute: perform the operation specified by the instruction.
- W Write: store the result in the destination location

8. What are Hazards?

There are situations in pipelining when the next instruction cannot execute in the following clock cycle. These events are called hazards, and there are three different types. Structural Hazards, Data Hazards & Control Hazards. Stall is introduced by hazard.

9. What is locality of reference?

Many instruction in localized area of the program are executed repeatedly during some time period and the remainder of the program is accessed relatively infrequently .this is referred as locality of reference.

10. Define memory access time?

The time that elapses between the initiation of an operation and completion of that operation, for example, the time between the READ and the MFC signals .This is Referred to as memory access time.

11) What is Datapath Element?

A unit used to operate on or hold data within a processor. In the MIPS implementation the datapath elements include the instruction and data memories, the register file, the ALU and adders.

12) Define Register File?

The processor's 32 general-purpose registers are stored in a structure called a register file. A register file is a collection of registers in which any register can be read or written by specifying the number of the register in the file. The register file contains the register state of the computer.

13) Define Branch Target Address?

The address specified in a branch, which becomes the new program counter(PC) if the branch is taken. In the MIPS architecture the branch target is given by the sum of the offset field of the instruction and the address of the instruction following the branch.

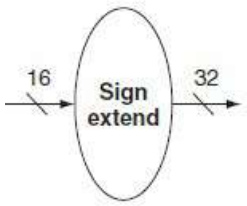
14) Define Branch Taken & Branch not taken?

A branch where the branch condition is satisfied and the program counter(PC) becomes the branch target. All unconditional branches are taken branches.

A branch where the branch condition is false and the program counter(PC) becomes the address of the instruction that sequentially follows the branch.

15) What is meant by sign extend?

To increase the size of a data item by replicating the high-order sign bit of the original data item in the high order bits of the larger, destination data item..



b. Sign-extension unit

16) What are the five steps in MIPS instruction execution?

1. Fetch instruction from memory.
2. Read registers while decoding the instruction. The regular format of MIPS instructions allows reading and decoding to occur simultaneously.
3. Execute the operation or calculate an address.
4. Access an operand in data memory.
5. Write the result into a register.

17) Define Forwarding?

Also called bypassing. A method of resolving a data hazard by retrieving the missing data element from internal buffers rather than waiting for it to arrive from programmer visible registers or memory.

18) What is pipeline stall?

Pipeline stall, also called bubble, is a stall initiated in order to resolve a hazard. They can be seen elsewhere in the pipeline.

19) How are jumps implemented?

We can implement a jump by storing into the PC the concatenation of

■ the upper 4 bits of the current PC + 4 (these are bits 31:28 of the sequentially following instruction address)

■ the 26-bit immediate field of the jump instruction

■ the bits 00two

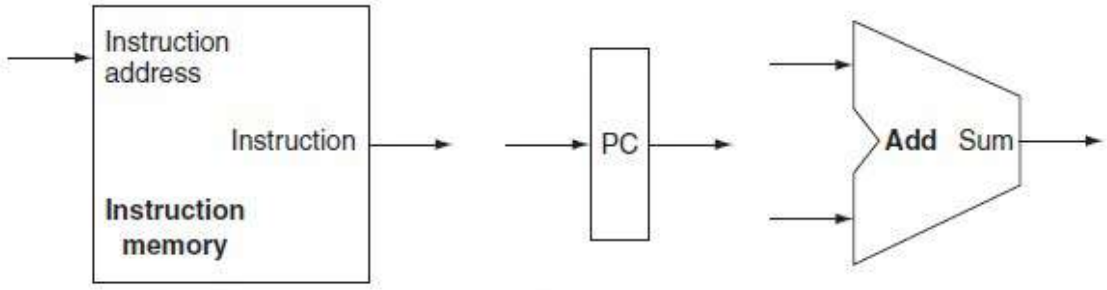
20) What are the 5 pipeline stages?

The 5 stages of instruction execution in a pipelined processor are: 1. IF: Instruction fetch 2. ID: Instruction decode and register file read 3. EX: Execution or address calculation 4. MEM: Data memory access 5. WB: Write back.

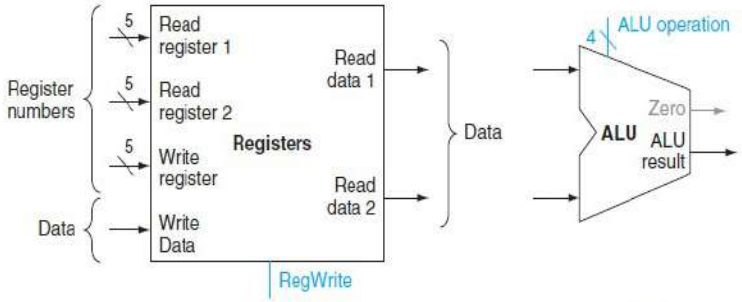
PART B

1. Explain data path and its control line in detail?

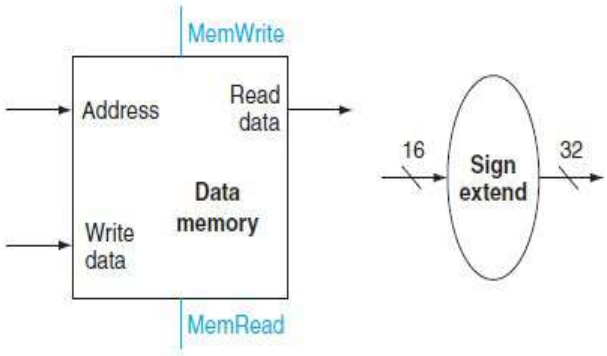
-Implementation of instructions (Arithmetic & logic, Memory reference, Branch Instruction)
-Program Counter- It holds the address of the current instruction
-Instruction Memory- It store the instructions of a program
-Adders- Used to add 32 bit inputs
-Register File: The processors 32 General Purpose Registers are stored here.
-ALU-



a. Instruction memory b. Program counter c. Adder



a. Registers b. ALU



a. Data memory unit b. Sign-extension unit

-Data Memory-The memory unit is a state element with inputs for the address and the write data, and a single output for the read result.
-Sign Extend- It increases the size of a data item by replicating the high-order sign bit of original data item
-Multiplexers

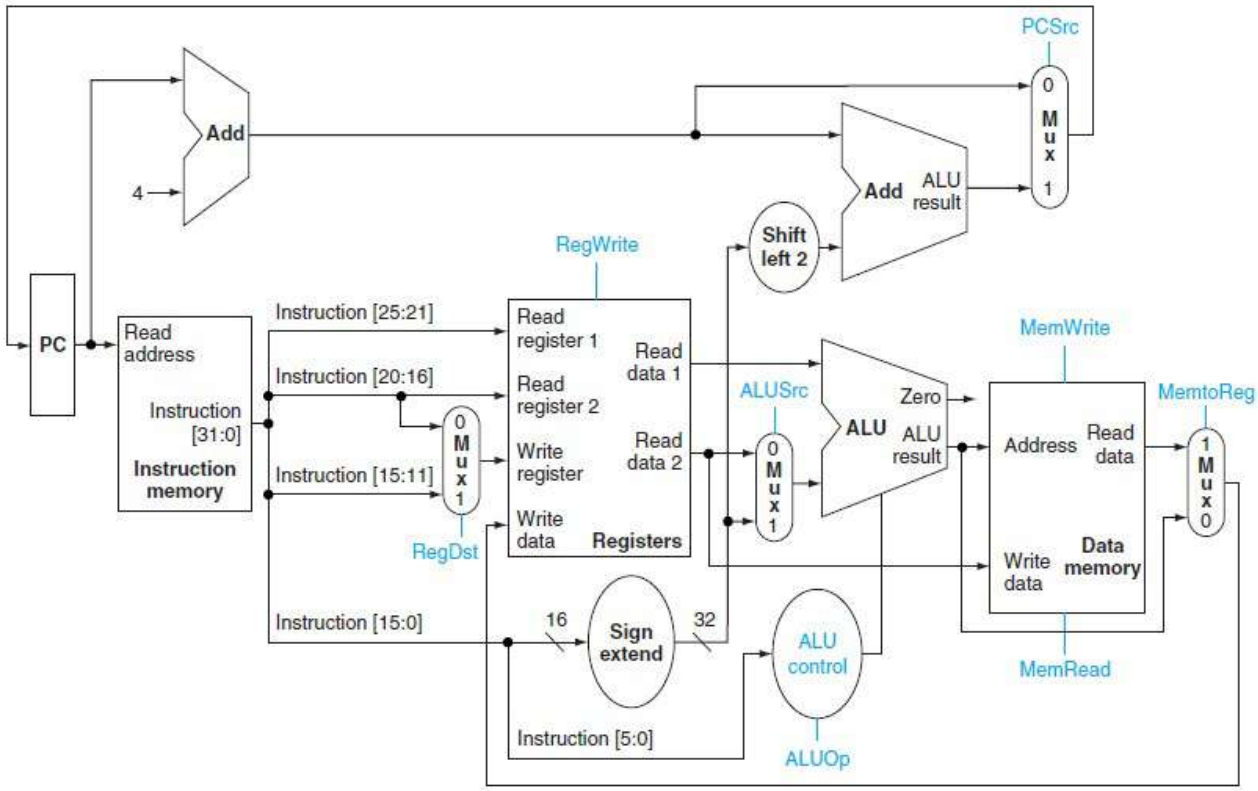


FIGURE 5.15 The datapath of Figure 5.12 with all necessary multiplexors and all control lines identified. The control lines are shown in color. The ALU control block has also been added. The PC does not require a write control, since it is written once at the end of every clock

Designing the main control unit

Field	0	rs	rt	rd	shamt	funct
Bit positions	31:26	25:21	20:16	15:11	10:6	5:0

a. R-type instruction

Field	35 or 43	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

b. Load or store instruction

Field	4	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

c. Branch instruction

FIGURE 5.14 The three instruction classes (R-type, load and store, and branch) use two different instruction formats. The jump instructions use another format, which we will discuss shortly. (a) Instruction format for R-format instructions, which all have an opcode of 0. These instructions have three register operands: rs, rt, and rd. Fields rs and rt are sources, and rd is the destination. The ALU function is in the funct field and is decoded by the ALU control design in the previous section. The R-type instructions that we implement are `add`, `sub`, `and`, `or`, and `slt`. The shamt field is used only for shifts; we will ignore it in this chapter. (b) Instruction format for load (opcode = 35_{ten}) and store (opcode = 43_{ten}) instructions. The register rs is the base register that is added to the 16-bit address field to form the memory address. For loads, rt is the destination register for the loaded value. For stores, rt is the source register whose value should be stored into memory. (c) Instruction format for branch equal (opcode = 4). The registers rs and rt are the source registers that are compared for equality. The 16-bit address field is sign-extended, shifted, and added to the PC to compute the branch target address.

Signal name	Effect when deasserted	Effect when asserted
RegDst	The register destination number for the Write register comes from the rt field (bits 20:16).	The register destination number for the Write register comes from the rd field (bits 15:11).
RegWrite	None.	The register on the Write register input is written with the value on the Write data input.
ALUSrc	The second ALU operand comes from the second register file output (Read data 2).	The second ALU operand is the sign-extended, lower 16 bits of the instruction.
PCSrc	The PC is replaced by the output of the adder that computes the value of PC + 4.	The PC is replaced by the output of the adder that computes the branch target.
MemRead	None.	Data memory contents designated by the address input are put on the Read data output.
MemWrite	None.	Data memory contents designated by the address input are replaced by the value on the Write data input.
MemtoReg	The value fed to the register Write data input comes from the ALU.	The value fed to the register Write data input comes from the data memory.

FIGURE 5.16 The effect of each of the seven control signals. When the 1-bit control to a two-way multiplexor is asserted, the multiplexor selects the input corresponding to 1. Otherwise, if the control is deasserted, the multiplexor selects the 0 input. Remember that the state elements all have the clock as an

-Operations of the datapath for an R-type instruction.

- Operations of the datapath for a load instruction

-Operations of the datapath for a branch on equal instruction

2. Explain different type of pipeline hazards with suitable example?

There are situations in pipelining when the next instruction cannot execute in the following clock cycle. These events are called *hazards*, and there are three different types.

#Structural Hazards

The first hazard is called a **structural hazard**. It means that the hardware cannot support the combination of instructions that we want to execute in the same clock cycle. The MIPS instruction set was designed to be pipelined, making it fairly easy for designers to avoid structural hazards when designing a pipeline.

#Data Hazards

Data hazards occur when the pipeline must be stalled because one step must wait for another to complete. In a computer pipeline, data hazards arise from the dependence of one instruction on an earlier one that is still in the pipeline.

The primary solution is based on the observation that we don't need to wait for the instruction to complete before trying to resolve the data hazard. Adding extra hardware to retrieve the missing item early from the internal resources is called **forwarding** or **bypassing**.

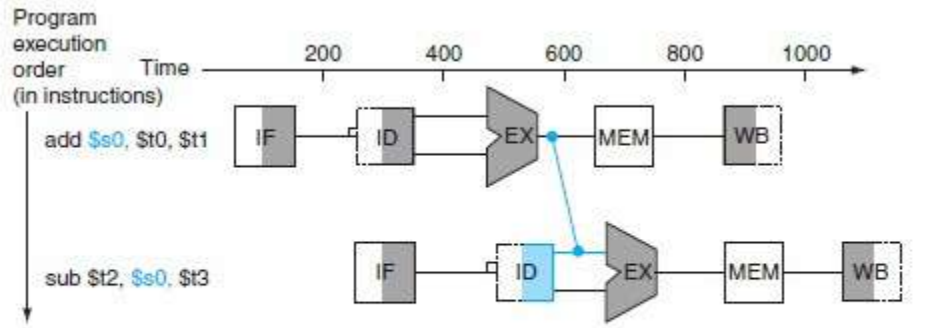
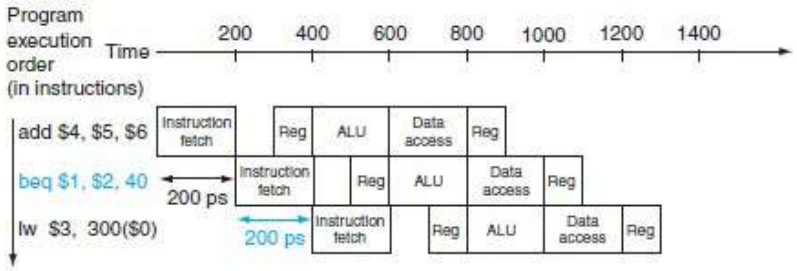


FIGURE 6.5 Graphical representation of forwarding. The connection shows the forwarding path from the output of the EX stage of `add` to the input of the EX stage for `sub`, replacing the value from register `$s0` read in the second stage of `sub`.

Control Hazards

The third type of hazard is called a **control hazard**, arising from the need to make a decision based on the results of one instruction while others are executing. Computers use *prediction* to handle branches. One simple approach is to always predict that branches will be **untaken**. When you're right, the pipeline proceeds at full speed. Only when branches are taken does the pipeline stall.



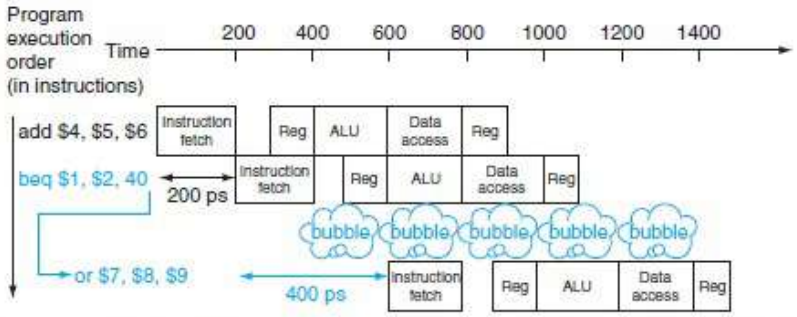


FIGURE 6.8 Predicting that branches are not taken as a solution to control hazard. The

Other approach is to look up the address of the instruction to see if a branch was taken the last time this instruction was executed, and, if so, to begin fetching new instructions from the same place as the last time. This technique is called **dynamic branch prediction**. One implementation of that approach is a **branch prediction buffer** or **branch history table**. A branch prediction buffer is a small memory indexed by the lower portion of the address of the branch instruction. The memory contains a bit that says whether the branch was recently taken or not.

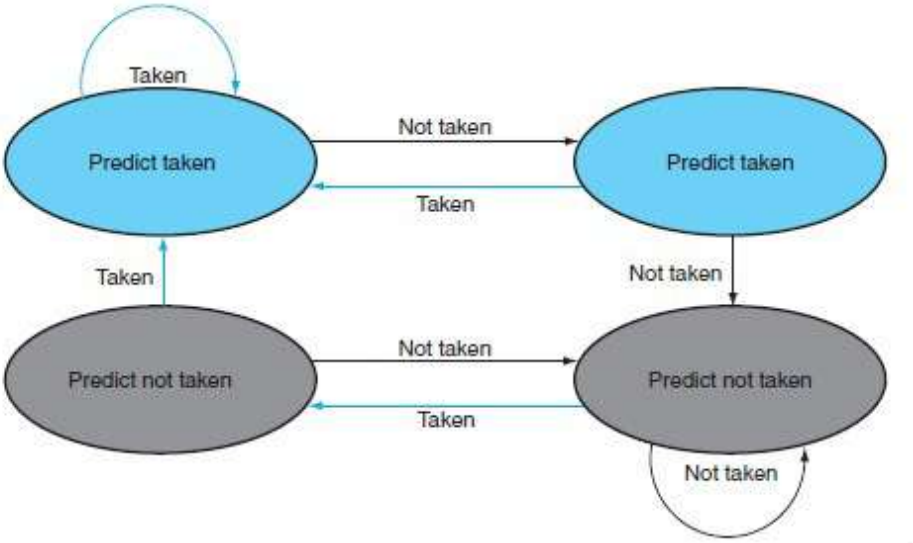


FIGURE 6.39 The states in a 2-bit prediction scheme. By using 2 bits rather than 1, a branch that strongly favors taken or not taken—as many branches do—will be mispredicted only once. The 2 bits are used to encode the four states in the system. The two-bit scheme is a general instance of a counter-based predictor, which is incremented when the prediction is accurate and decremented otherwise, and uses the midpoint of its range as the division between taken and not taken.

4. Explain in detail how exception are handled n MIPS Architecture?

Another form of control hazard involves exceptions. For example, suppose the following instruction add $1,$2,$1 has an arithmetic overflow.

We need to transfer control to the exception routine immediately after this instruction because we wouldn't want this invalid value to contaminate other registers or memory locations.

We must flush the instructions that follow the add instruction from the pipeline and begin fetching instructions from the new address.

Flush signal is used to prevent the instruction in the EX stage from writing its result in the WB stage. We eventually have to complete the instruction that caused the exception as if it executed normally.

To do this flush the instruction and restart it from the beginning after the exception is handled. The final step is to save the address of the offending instruction in the Exception Program Counter (EPC),

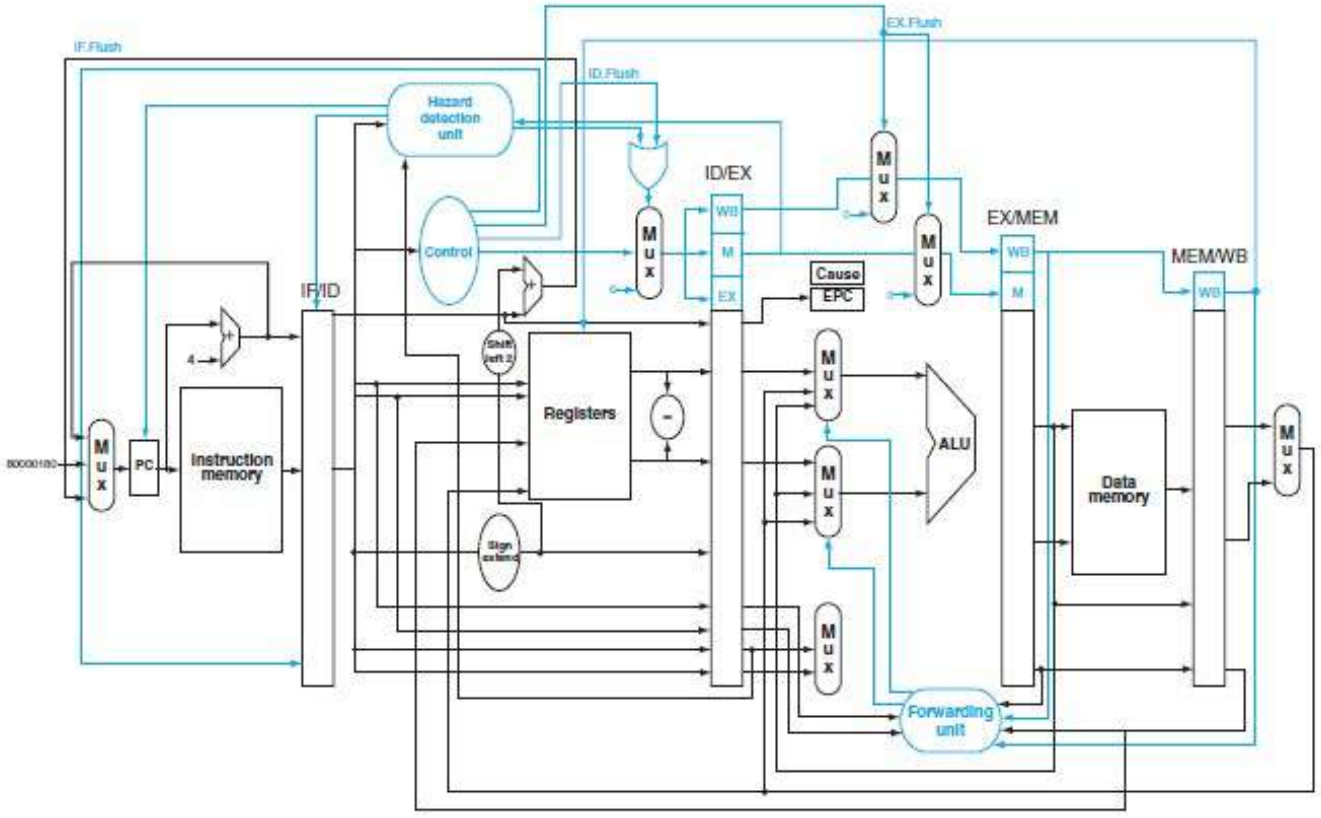


FIGURE 6.42 The datapath with controls to handle exceptions. The key additions include a new input, with the value 8000 0180$_{hex}$, in the multiplexor that supplies the new PC value; a Cause register to record the cause of the exception; and an Exception PC register to save the address of the instruction that caused the exception. The 8000 0180$_{hex}$ input to the multiplexor is the initial address to begin fetching instructions in the event of an exception. Although not shown, the ALU overflow signal is an input to the control unit.

5. What is pipelining? Discuss about pipeline data path control?

The division of an instruction into five stages means a five-stage pipeline, which in turn means that up to five instructions will be in execution during any single clock cycle.

We can separate the datapath into five pieces, with each piece named corresponding to a stage of instruction execution:

1. IF: Instruction fetch.
2. ID: Instruction decode and register file read
3. EX: Execution or address calculation
4. MEM: Data memory access
5. WB: Write back

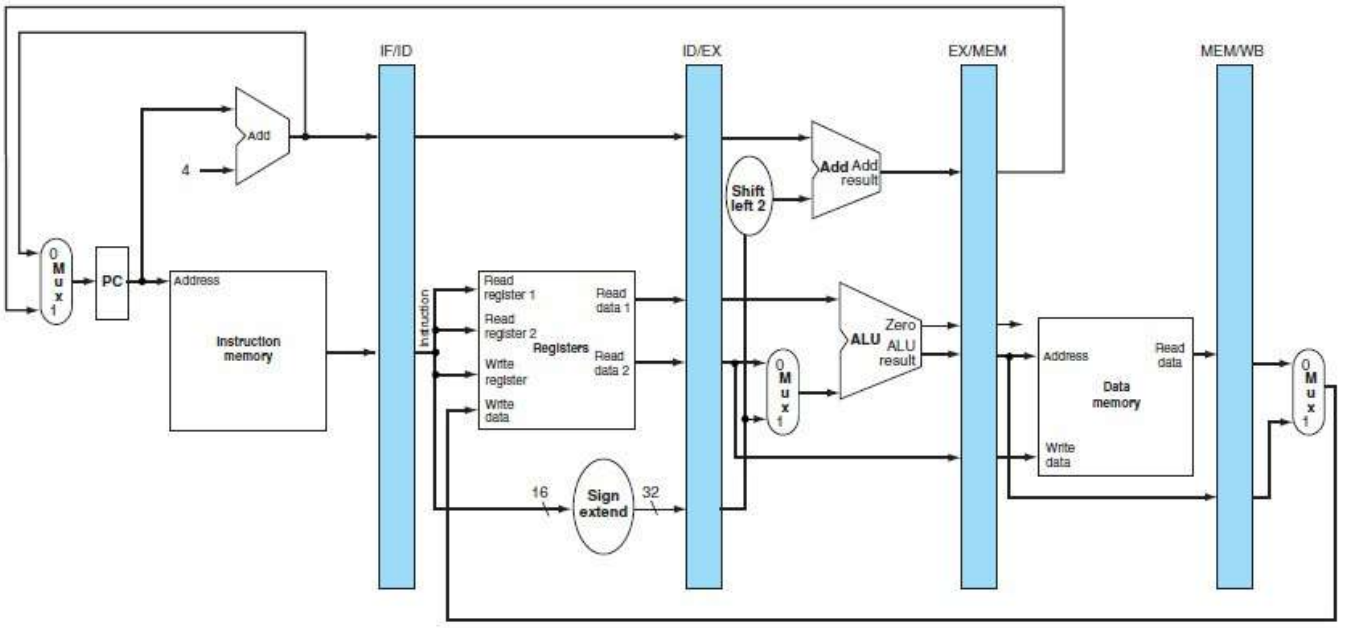


FIGURE 6.11 The pipelined version of the datapath in Figure 6.9. The pipeline registers, in color, separate each pipeline stage. They are

Many instructions are simultaneously executing in a single datapath in every clock cycle

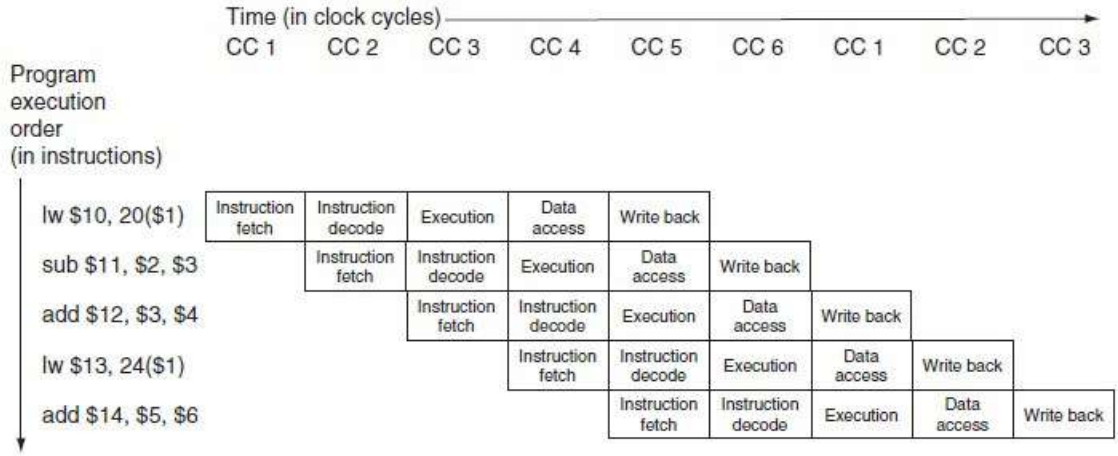


FIGURE 6.20 Traditional multiple-clock-cycle pipeline diagram of five instructions in Figure 6.19.

QUESTION BANK – UNIT IV

2 MARKS

1. What is Instruction Level Parallelism?

Pipelining is used to overlap the execution of instructions and improve performance. This potential overlap among instructions is called instruction level parallelism (ILP).

2. What is Multithreading?

Multithreading allows multiple threads to share the functional units of a single processor in an overlapping fashion. To permit this sharing, the processor must duplicate the independent state of each thread.

3. What is meant by hardware multithreading?

Hardware multithreading allows multiple threads to share the functional units of a single processor in an overlapping fashion to try to utilize the hardware resources efficiently. To permit this sharing, the processor must duplicate the independent state of each thread. It Increases the utilization of a processor.

4. What is Flynn's classification?

SISD – Single Instruction Single DataStream

SIMD – Single Instruction Multiple DataStream

MISD – Multiple Instruction Multiple DataStream

MIMD – Multiple Instruction Multiple DataStream

5. Differentiate between Strong Scaling and Weak Scaling?

Strong scaling: Measuring speedup by increasing the size of the problem

Weak scaling: The program size grows proportionally to the increase in the number of processors

6. Compare UMA and NUMA multiprocessors?

UMA: Multiprocessors take about same time to access main memory no matter which processors request it and no matter which word is requested.

NUMA: Some memory accesses are much faster than others, depending on which processors ask for the word.

7. What is fine grained multithreading?

This multithreading switch between threads on each instruction, resulting in interleaved execution of multiple treads

8. Define a super scalar processor?

It is a CPU that implements a form of parallelism called Instruction level parallelism with a single processor.

It executes more than one instruction in a given clock cycle

9. What is multicore'?

At its simplest, multi-core is a design in which a single physical processor contains the core logic of more than one processor.

The multi-core design takes several such processor "cores" and packages them as a single physical processor. The goal of this design is to enable a system to run more tasks simultaneously and thereby achieve greater overall system performance.

10. What is multiple issue? Write any two approaches.

Multiple issues is a scheme whereby multiple instructions are launched in one clock cycle. It is a method for increasing the potential amount of instruction-level parallelism. It is done by replicating the internal components of the computer so that it can launch multiple instructions in every pipeline stage.

The two approaches are:

1. Static multiple issue (at compile time)

2. Dynamic multiple issue (at run time)

11. What are Superscalar processors?

It is an advanced pipelining technique that enables the processor to execute more than one instruction per clock cycle.

12. Define a reservation station?

It is a buffer within a functional unit that holds the operands and the operation.

13. Explain dynamic pipeline scheduling.

Here with the hardware support we perform the reordering of the order of instruction execution so as to avoid stalls.

14. Explain out of order execution.

It is a situation in pipelined execution when an instruction blocked from executing does not cause the following instructions to wait.

15. Define a Thread.

A thread includes the program counter, the register state and stack. It is a lightweight process. The commonly share a single address space.

16. Define a process.

A process includes one or more threads, the address space and the operating system state. During a process switch operating system is invoked.

PART B

1. Explain Instruction level parallel processing? State the challenges of Parallel processing?

Pipelining exploits the potential parallelism among instructions. This parallelism is called **instruction-level parallelism** (ILP). There are two primary methods for increasing the potential amount of instruction-level parallelism.

The first is increasing the depth of the pipeline to overlap more instructions.

We would then move from a four-stage to a six stage pipeline. The amount of parallelism being exploited is higher, since there are more operations being overlapped. Performance is potentially greater since the clock cycle can be shorter.

Another approach is to replicate the internal components of the computer so that it can launch multiple instructions in every pipeline stage. The general name for this technique is **multiple issue**. There are two primary and distinct responsibilities that must be dealt with in a multiple-issue pipeline.

Packaging instructions into **issue slots**

Dealing with data and control hazards:

Speculation is an approach that allows the compiler or the processor to "guess" about the properties of an instruction, so as to enable execution to begin for other instructions that may depend on the speculated instruction

Static Multiple Issue

Static multiple-issue processors all use the compiler to assist with packaging instructions and handling hazards. In a static issue processor, you can think of the set of instructions that issue in a given clock cycle, which is called an **issue packet**, as one large instruction with multiple operations.

Instruction type	Pipe stages							
ALU or branch instruction	IF	ID	EX	MEM	WB			
Load or store instruction	IF	ID	EX	MEM	WB			
ALU or branch instruction		IF	ID	EX	MEM	WB		
Load or store instruction		IF	ID	EX	MEM	WB		
ALU or branch instruction			IF	ID	EX	MEM	WB	
Load or store instruction			IF	ID	EX	MEM	WB	
ALU or branch instruction				IF	ID	EX	MEM	WB
Load or store instruction				IF	ID	EX	MEM	WB

FIGURE 6.44 Static two-issue pipeline in operation. The ALU and data transfer instructions are issued at the same time. Here we have assumed the same five-stage structure as used for the single-issue pipeline. Although this is not strictly necessary, it does have some advantages. In particular, keeping the register writes at the end of the pipeline simplifies the handling of exceptions and the maintenance of a precise exception model, which become more difficult in multiple-issue processors.

Dynamic Multiple-Issue Processors

Dynamic multiple-issue processors are also known as **superscalar** processors, or simply superscalars. In the simplest superscalar processors, instructions issue inorder, and the processor decides whether zero, one, or more instructions can issue in a given clock cycle. Dynamic pipeline scheduling chooses which instructions to execute in a given clock cycle while trying to avoid hazards and stalls. Let's start with a simple example of avoiding a data hazard. Consider the following code sequence:

```
lw $t0, 20($s2)
addu $t1, $t0, $t2
sub $s4, $s4, $t3
slti $t5, $s4, 20
```

Even though the sub instruction is ready to execute, it must wait for the lw and addu to complete first, which might take many clock cycles if memory is slow.

Dynamic Pipeline Scheduling

Dynamic pipeline scheduling chooses which instructions to execute next, possibly reordering them to avoid stalls.

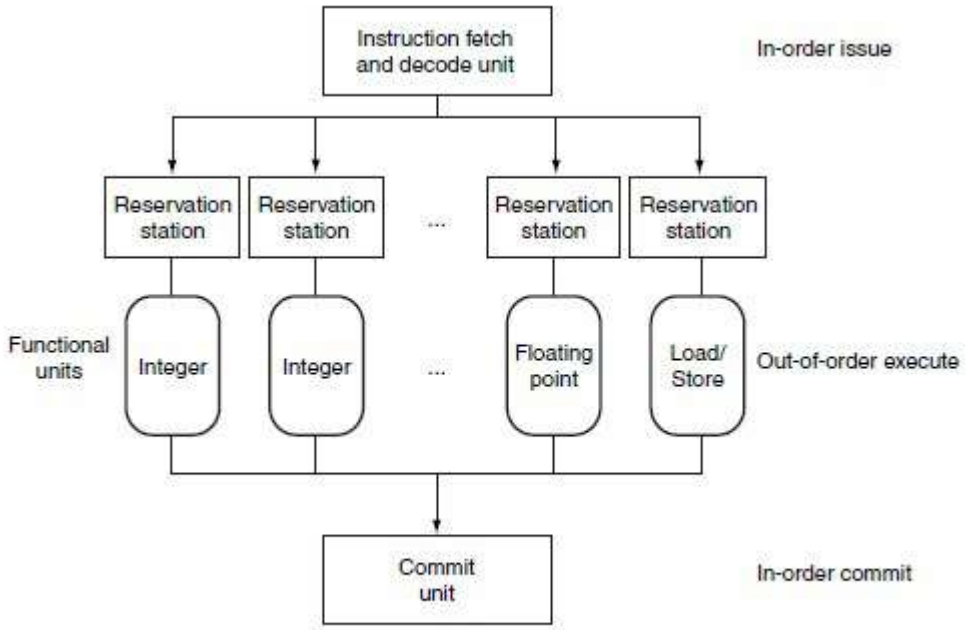


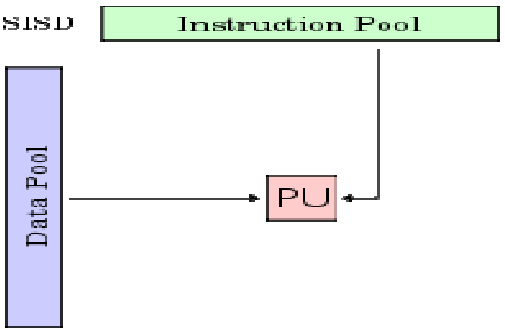
FIGURE 6.49 The three primary units of a dynamically scheduled pipeline. The final step of updating the state is also called retirement or graduation.

2. Explain Flynn's classification in detail? Or Discuss about SISD, SIMD, MISD, MIMD ?

SISD (Single Instruction, Single Data stream)

It is an Instruction Set Architecture in which a single processor (one CPU) executes exactly one instruction stream at a time.

It also fetches or stores one item of data at a time to operate on data stored in a single memory unit.

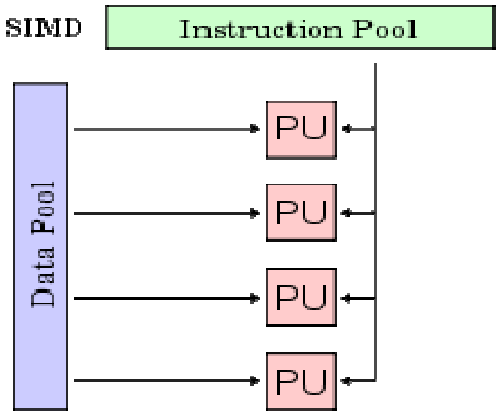


SIMD (Single Instruction, Multiple Data streams)

It is an Instruction Set Architecture that have a single control unit (CU) and more than one processing unit (PU).

It operates by executing a single instruction stream over multiple PU's.

The CU generates the control signals for all of the PUs and by which executes the same operation on different data streams.

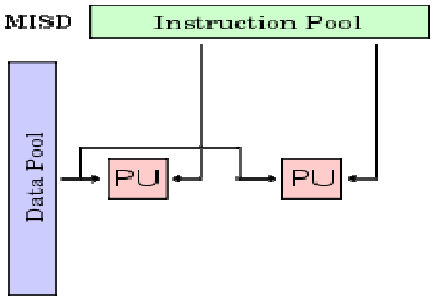


MISD (Multiple Instruction, Single Data stream)

It is an Instruction Set Architecture for parallel computing .

Here many functional units perform different operations by executing different instructions on the same data set.

These are mainly used in the fault-tolerant computers.

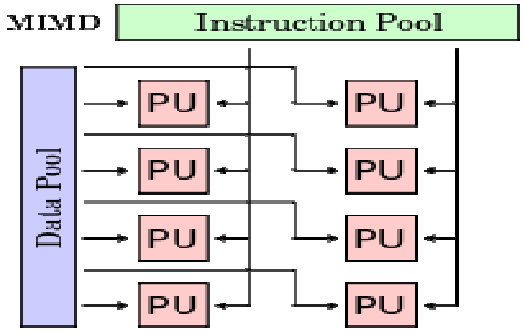


MIMD(Multiple Instruction, Multiple Data streams)

It is an Instruction Set Architecture for parallel computing usually used in computers with multiprocessors.

Here, each processor in a multiprocessor system can execute asynchronously different set of the instructions independently on the different set of data units.

The MIMD based computer systems can used the shared memory in a memory pool or work using distributed memory.

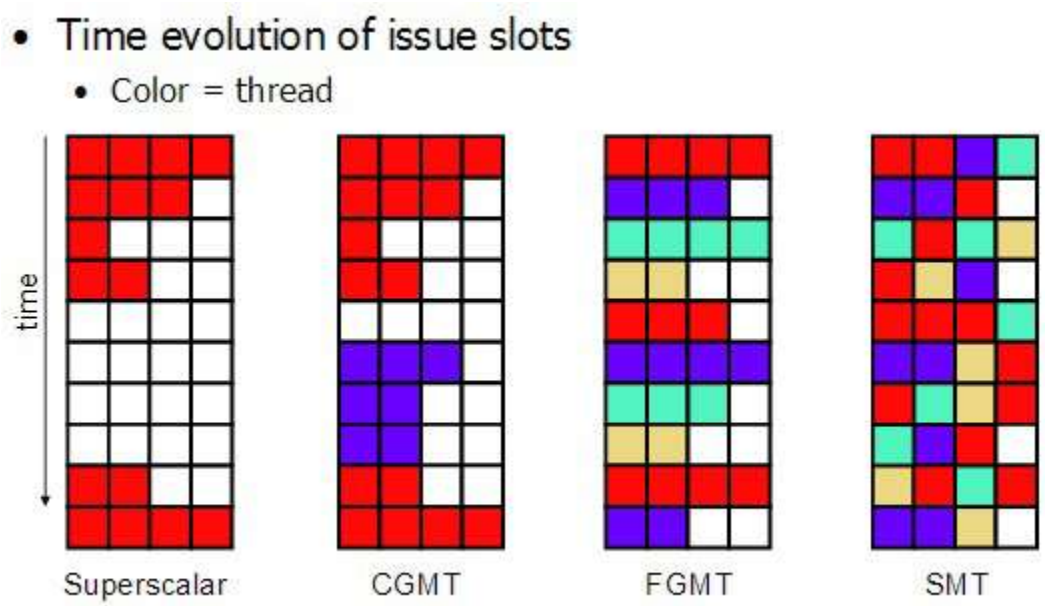


3. What is Hardware multithreading? Explain its type in detail?

Multi-threading (MT)

Multithreading allows overlap of memory access of one thread/process with computation by another thread/process.
It improve utilization by multiplexing multiple threads on single core.
If one thread cannot fully utilize core we can have more than one thread.

Three designs
• Coarse-grain multithreading (CGMT)
• Fine-grain multithreading (FGMT)
• Simultaneous multithreading (SMT)



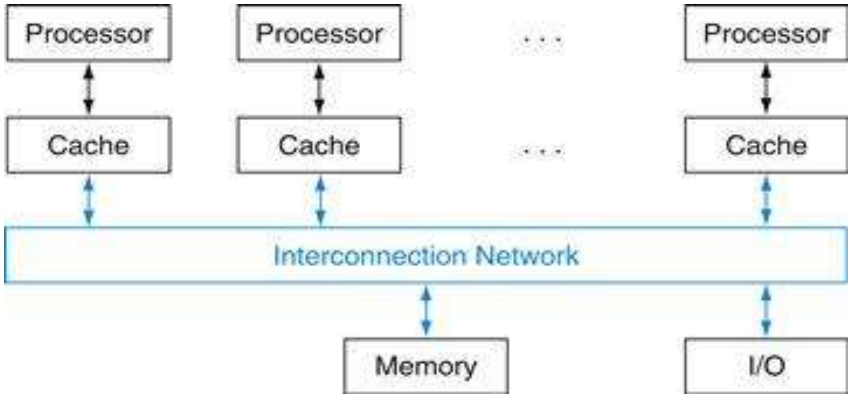
Coarse-Grain Multi-Threading (CGMT)
It follows Priority thread scheduling policy
It designates a "preferred" thread (e.g., thread A) and then switch to thread B on thread A cache miss again switch back to A when A cache miss returns.
Example: IBM Northstar/Pulsar

Fine-Grain Multithreading (FGMT)
If we have so many threads that never stall on cache misses then we have to apply Thread scheduling policy
Here we Switch threads every cycle (round-robin). Here we need a lot of threads
Extreme example: Denelcor HEP, Tera MTA

Simultaneous Multithreading (SMT)
Here we can issue instructions from multiple threads in one cycle.
It needs more hardware support.
It maximizes utilization of execution units.
• IBM Power5: 4-way issue, 2 threads

5. Explain Multicore processors.

Processors communicate through shared variables in memory, with all processors capable of accessing any memory location via loads and stores



Uniform Memory Access (UMA) A multiprocessor in which latency to any word in main memory is about the same no matter which processor requests the access.

Nonuniform Memory Access (NUMA) A type of single address space multiprocessor in which some memory accesses are much faster than others depending on which processor asks for which word.

Synchronization The process of coordinating the behavior of two or more processes, which may be running on different processors.

Lock A synchronization device that allows access to data to only one processor at a time.

QUESTION BANK – UNIT V

2 MARKS

1. Differentiate Programmed I/O and Interrupt I/O?

Interrupt I/O is more efficient than Programmed I/O because it eliminates needless waiting

However Interrupt I/O consumes a lot of processors time

2. What is the purpose of dirty/modified bit in Cache memory?

To tackle whether a page has been written since it was read into the memory, o dirty but is added to the page table.

3. What is the need to implement memory as a hierarchy?

The aim of the memory hierarchy is to match the processors speed with the rate of information transfer at a lowest level and at a minimum possible cost

4. Point out how DMA can improve I/O speed.

DMA is implemented with a specialized controller that transfers data between an I/O device and memory independent of the processor

The DMA controller becomes the master and directs the reads and writes between itself and memory. During data transfer is not routed through the processors

5. Define Memory hierarchy?

1. Processor memory
2. Main memory
3. Secondary memory

6. State the advantages of virtual memory?

Virtual memory is feature of OS that allows a computer to compensate for shortage of physical memory by temporarily transferring pages of data from Random Access Memory to Disk

7. What are the various memory technologies?

Flash memory, RAM – Random Access Memory, ROM – Read Only Memory etc

8. Define Hit ratio?

Hit ratio or Hit rate is the fraction of memory accesses found in the upper level memory hierarchy.

Hit ratio is often used as a measure of the performance of the memory hierarchy.

9. What is Interrupt?

An interrupt is an unscheduled event that disrupts program execution, used to detect overflow

10. What is meant by Bus arbitration?

It is a way of sharing the computer's data transferring channels in an optimal way so that faster devices won't have to wait to be able to transfer and the slower devices will have a chance to transfer as well

1) What is Direct mapped cache?

A cache structure in which each memory location is mapped to exactly one location (address) in the cache. This done by (Block Address) modulo (Number of blocks in the cache).

2) Define Cache Miss?

A request for data from the cache that cannot be filled because the data is not present in the cache. The control unit must detect a miss and process the miss by fetching the requested data from memory.

3) Define Least Recently Used?

A replacement scheme in which the block replaced is the one that has been unused for the longest time.LRU replacement is implemented by keeping track of when each element in a set was used relative to the other elements in the set.

4) What is Virtual Memory?

A technique that uses main memory as a cache for secondary storage, usually implemented with magnetic disks. Virtual memory used to allow efficient and safe sharing of memory among multiple programs.

5) Define Segmentation?

A variable-size address mapping scheme in which an address consists of two parts: a segment number,which is mapped to a physical address, and a segment offset.

6) Define Translation-Lookaside Buffer (TLB)?

Most commercial virtual memory systems incorporate a mechanism that can avoid the bulk of the main memory access called for by the virtual to physical addresses translation buffer. This may be done with a cache memory called a translation buffer, which retains the results of the recent translation.

7) What is Polling?

The process of periodically checking the status of an I/O device to determine the need to service the device.

8) Define Direct Memory Access?

A mechanism that provides a device controller with the ability to transfer data directly to or from the memory without involving the processor.

9) What are the components of memory management unit?

1. A facility for dynamic storage relocation that maps logical memory references into physical memory addresses.
2. A provision for sharing common programs stored in memory by different users.
3. Protection of information against unauthorized access between users and preventing users from changing operating system functions.

10) Define latency time?

This is the amount of time that elapses after the head is positioned over the correct track until the starting position of the addressed sector passes under the read/write head.

PART B

1.Explain mapping function in cache memory to determine how memory blocks are placed in cache?

Cache Memories

The cache is a small and very fast memory, interposed between the processor and the main memory. Its purpose is to make the main memory appear to the processor to be much faster than it actually is.
There are several possible methods for determining where memory blocks are placed in the cache.

Direct Mapping

In this technique, block *j* of the main memory maps onto block *j* modulo 128 of the cache, as depicted in Figure 8.16.

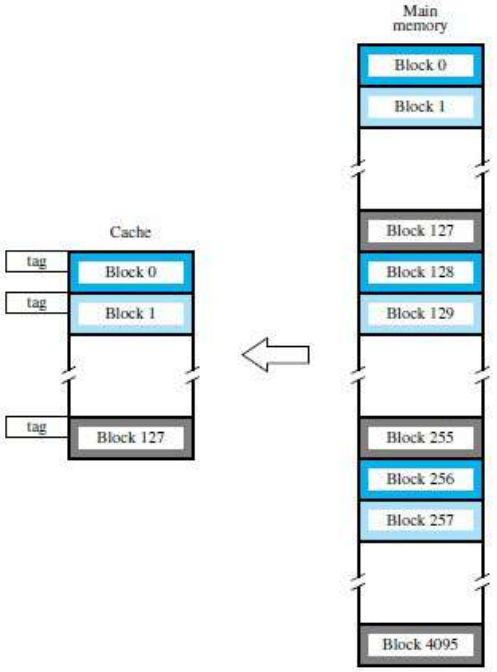


Figure 8.16 Direct-mapped cache.

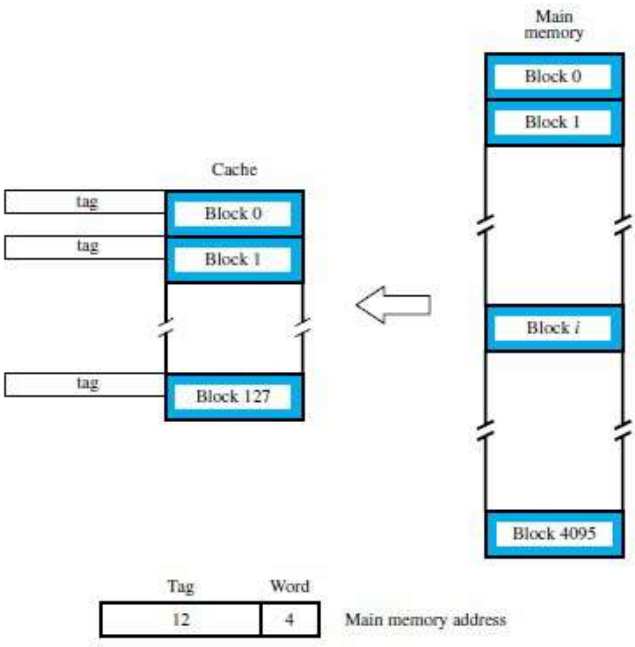


Figure 8.17 Associative-mapped cache.

Thus, whenever one of the main memory blocks 0, 128, 256, . . . is loaded into the cache, it is stored in cache block 0. Blocks 1, 129, 257, . . . are stored in cache block 1, and so on.

Associative-mapping

It is the most flexible mapping method, in which a main memory block can be placed into any cache block position.

Here, 12 tag bits are required to identify a memory block when it is resident in the cache.

It gives complete freedom in choosing the cache location in which to place the memory block, resulting in a more efficient use of the space in the cache

Set-Associative Mapping

Here we use a combination of the direct- and associative-mapping techniques. The blocks of the cache are grouped into sets, and the mapping allows a block of the main memory to reside in any block of a specific set. Figure 8.18 for a cache with two blocks per set.

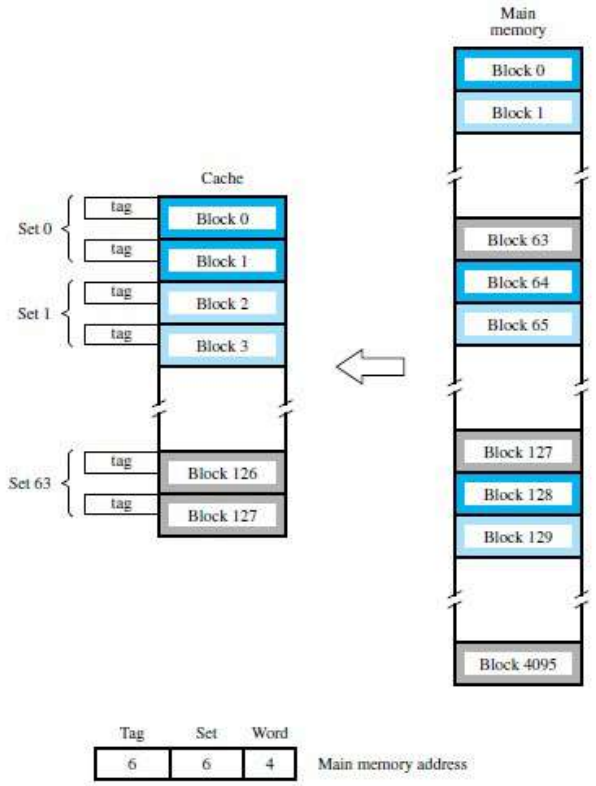


Figure 8.18 Set-associative-mapped cache with two blocks per set.

2.Explain in detail about bus Arbitration techniques in DMA?

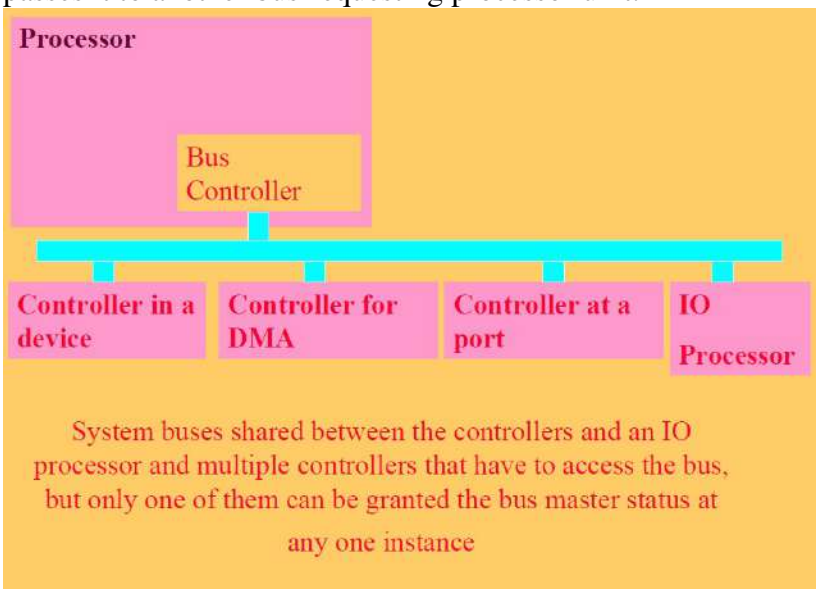
Trying to get access to a bus at the same time, but access can be given to only one of these.Only one processor or controller can be **bus master**

• The bus master─ it is the controller that has access to a bus at an instance

• Any one controller or processor can be the bus master at the given instance

Bus arbitration process

• Refers to a process by which the current bus master accesses and then leaves the control of the bus and passes it to another bus-requesting processor unit.



Three bus arbitration processes

1. Daisy Chain
2. Independent Bus Requests and Grant
3. Polling

Daisy Chain Method

• A method for a centralized bus arbitration process

• The bus control passes from one bus master to the next one, then to the next and so on

Independent bus request method

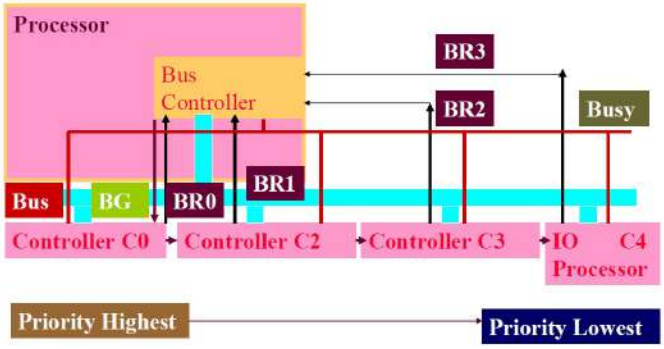
_ Controller separate BR signals, BR0, BR1,…, BRn.

_ Separate BG signals, BG0, BG1, …, BGn for the controllers.

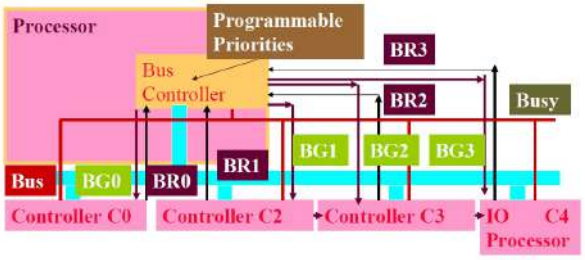
Any controller, which finds active BUSY, does not send BR from it.

Bus polling method

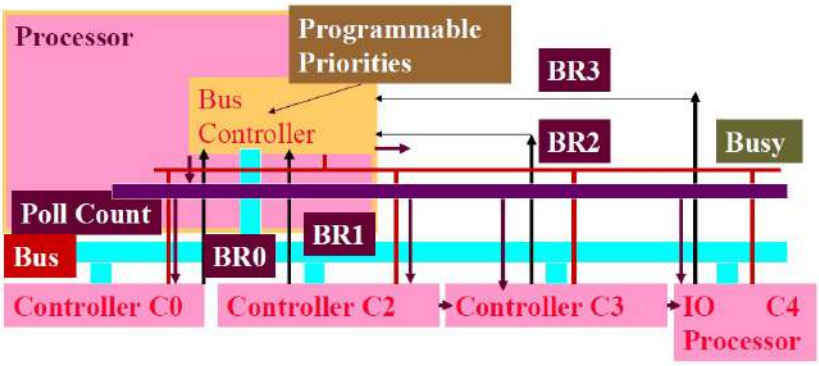
• The bus control passes from one processor (bus controller) to another only through the centralized bus controller, but only when the controller sends poll count bits, which correspond to the unit number.



Daisy Chaining



Independent request and grant



Polling Method

3. Elaborate on the various memory technologies and its relevance?

Static Memories

Static RAMs can be accessed very quickly. Continuous power is needed for the cell to retain its state. If power is interrupted, the cell's contents are lost.

Dynamic RAMs

DRAM cells do not retain their state for a long period, unless they are accessed frequently for Read or Write operations. Memories that use such cells are called *dynamic RAMs* (DRAMs)

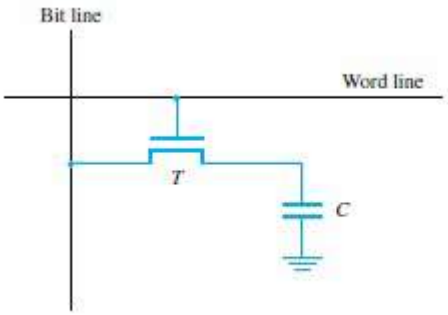


Figure 8.6 A single-transistor dynamic memory cell.

Synchronous DRAMs

DRAMs operation is synchronized with a clock signal. Such memories are known as *synchronous DRAMs* (SDRAMs).

DDR SDRAMs

To make the best use of the available clock speed, data are transferred externally on both the rising and falling edges of the clock. For this reason, memories that use this technique are called *double-data-rate SDRAMs* (DDR SDRAMs).

Rambus Memory

Rambus is a memory technology that achieves a high data transfer rate by providing a high-speed interface between the memory and the processor. One way for increasing the bandwidth of this connection is to use a wider data path. However, this requires more space and more pins, increasing system cost. The alternative is to use fewer wires with a higher clock speed. This is the approach taken by Rambus.

4. What is Virtual memory? Explains the steps involved in virtual memory address translation?

Virtual memory as an alternate set of memory addresses.

Programs use these virtual addresses rather than real addresses to store instructions and data.

When the program is actually executed, the virtual addresses are converted into real memory addresses.

When a computer is executing many programs at the same time, Virtual memory make the computer to share memory efficiently.

To facilitate copying virtual memory into real memory, the operating system divides virtual memory into pages, each of which contains a fixed number of addresses.

Each page is stored on a disk until it is needed.

When the page is needed, the operating system copies it from disk to main memory, translating the virtual addresses into real addresses.

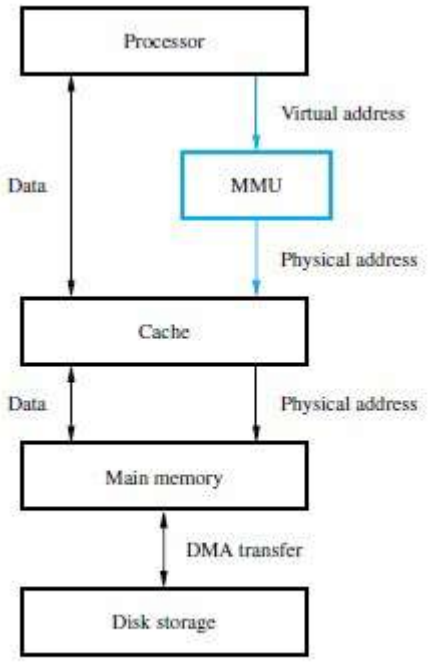


Figure 8.24 Virtual memory organization.

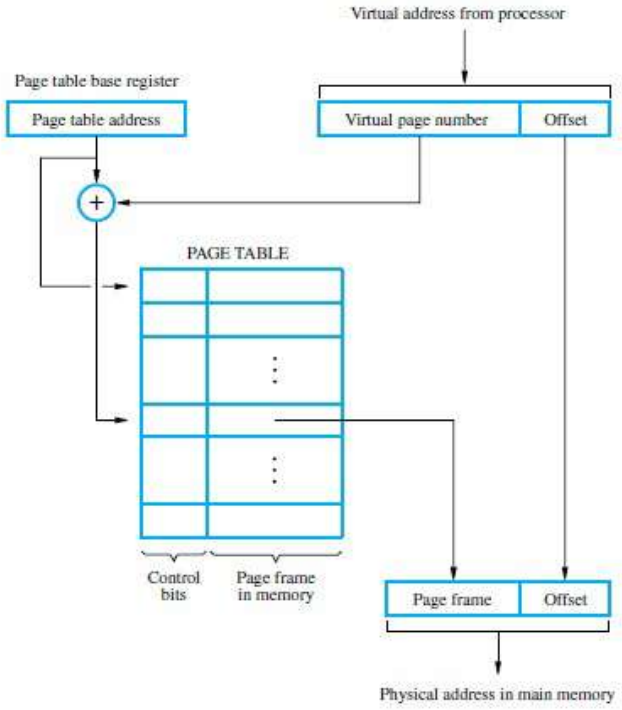


Figure 8.25 Virtual-memory address translation.

- MMU is the hardware base that makes a virtual memory system possible.

MMU allows software to reference physical memory by virtual addresses, quite often more than one.It accomplishes this through the use of page and page tables.

Paging is a technique used by virtual memory operating systems to help ensure that the data you need is available as quickly as possible.

In paging the operating system copies a certain number of pages from your storage device to main memory.

When a program needs a page that is not in maim memory, the operating system copies the required page into memory and copies another page back to the disk.

TLBs

A way to speed up translation is to use a special cache of recently used page table entries -- this has many names, but the most frequently used is *Translation Lookaside Buffer* or *TLB*

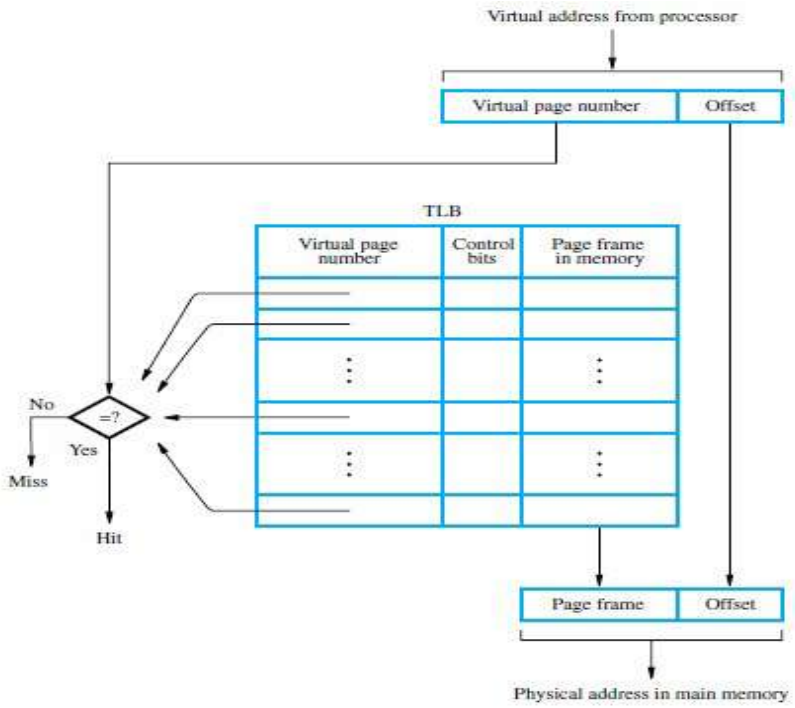


Figure 8.26 Use of an associative-mapped TLB.

6. Explain about DMA controller with the help of neat diagram?

DMA based method useful, when a block of bytes are transferred, for example, from disk to the RAM or RAM to the disk.

Repeatedly interrupting the processor for transfer of every byte during bulk transfer of data will waste too much of processor time in context switching.

System performance improves by separate processing of the transfers from and to the peripherals.

DMA controller operates under the control of an operating system routine. To initiate the transfer of a block of words, the processor sends to the DMA controller the starting address, the number of words in the block, and the direction of the transfer.

The DMA controller then proceeds to perform the requested operation. When the entire block has been transferred, it informs the processor by raising an interrupt.

Figure 8.12 shows an example of the DMA controller registers that are accessed by the processor to initiate data transfer operations.

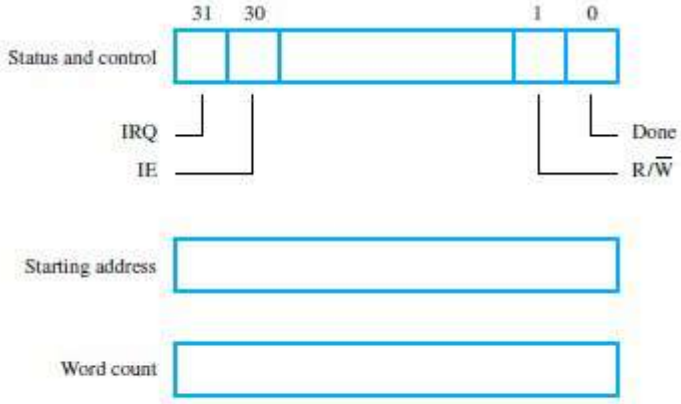


Figure 8.12 Typical registers in a DMA controller.

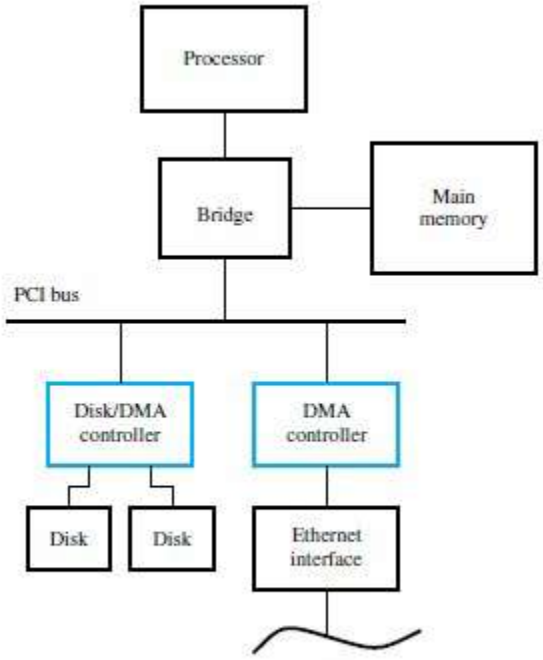


Figure 8.13 Use of DMA controllers in a computer system.

Figure 8.13 shows how DMA controllers may be used in a computer system