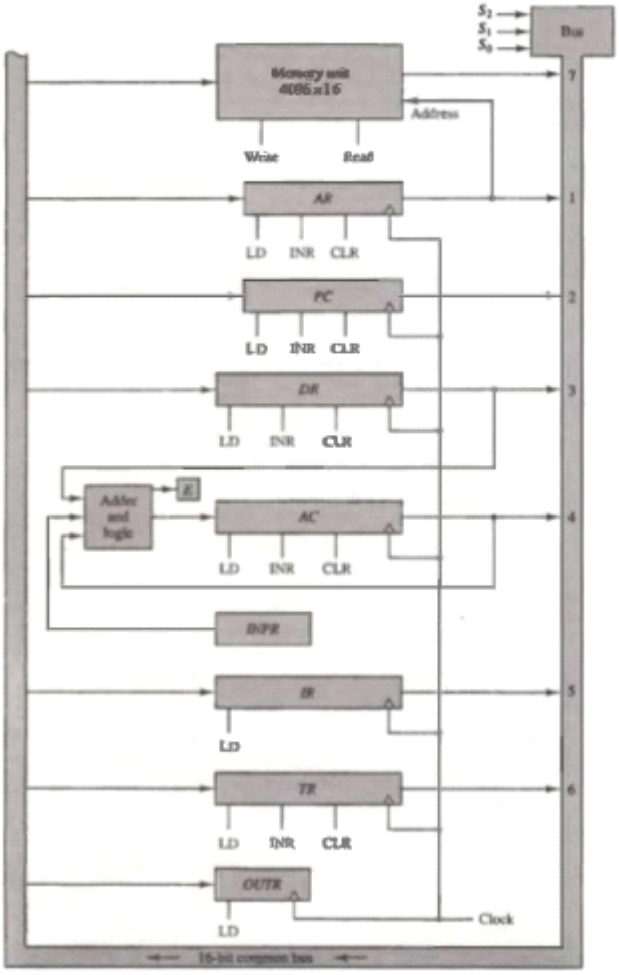
M. L. V. Govt. Textile Engineering College, Bhilwara Pur Road, Pratap Nagar
Marks 10 Subject (Computer architecture) (V Semester 2017-2018) Time 1 hr

Q.1Design and explain common bus system for eight registers.

Ans: The basic computer has eight registers, a memory unit, and a control unit Paths must be provided to transfer information from one register to another and between memory and registers. The number of wires will be excessive if connections are made between the outputs of each register and the inputs of the other registers. A more efficient scheme for transferring information in a system with many registers is to use a common bus.
The outputs of seven registers and memory are connected to the common bus. The specific output that is selected for the bus lines at any given time is determined from the binary value of the selection variables S2, S1, and 50• The number along each output shows the decimal equivalent of the required binary selection. For example, the number along the output of DR is 3. The 16-bit outputs of DR are placed on the bus lines when S2S1S0 = 011 since this is the binary value of decimal3. The lines from the common bus are connected to the inputs of each register and the data inputs of the memory. The particular register whose LD (load) input is enabled receives the data from the bus during the next clock pulse transition. The memory receives the contents of the bus when its write input is activated. The memory places its 16-bit output onto the bus when the read input is activated and S2S1S0 = 111



Q2) Explain binary adder.

Ans: To implement the add microoperation with hardware, we need the registers that hold the data and the digital component that performs the arithmetic addition. The digital circuit that forms the arithmetic sum of two bits and a previous carry is called a full-adder (see Fig. 1-17). The digital circuit that generates the arithmetic sum of two binary numbers of any length is called a binary adder. The binary adder is constructed with full-adder circuits connected in cascade, with the output carry from one full-adder connected to the input carry of the next full-adder. Figure 4-6 shows the interconnections of four full-adders (FA) to provide a 4-bit binary adder. The augend bits of A and the addend bits of B are designated by subscript numbers from right to left, with subscript 0 denoting the low-order bit. The carries are connected in a chain through the full-adders. The input carry to the binary adder is C0 and the output carry is C,. The S outputs of the full-adders generate the required sum bits. An n-bit binary adder requires n full-adders. The output carry from each full-adder is connected to the input carry of the next-high-order full-adder. The n data bits for the A inputs come from one register (such as R1), and the n data bits for the B inputs come from another register (such as R2). The sum can be transferred to a third register or to one of the source registers (R 1 or R2), replacing its previous content.

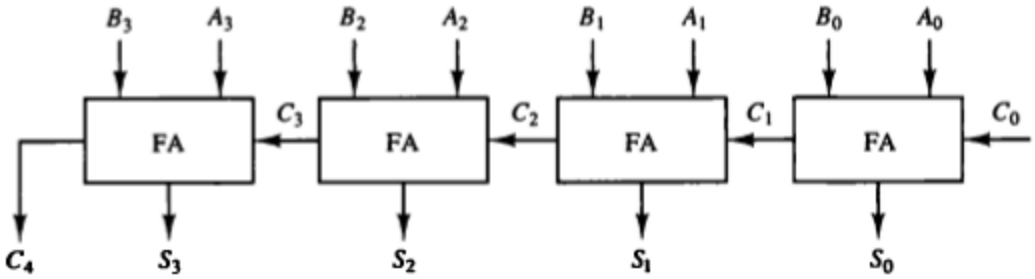


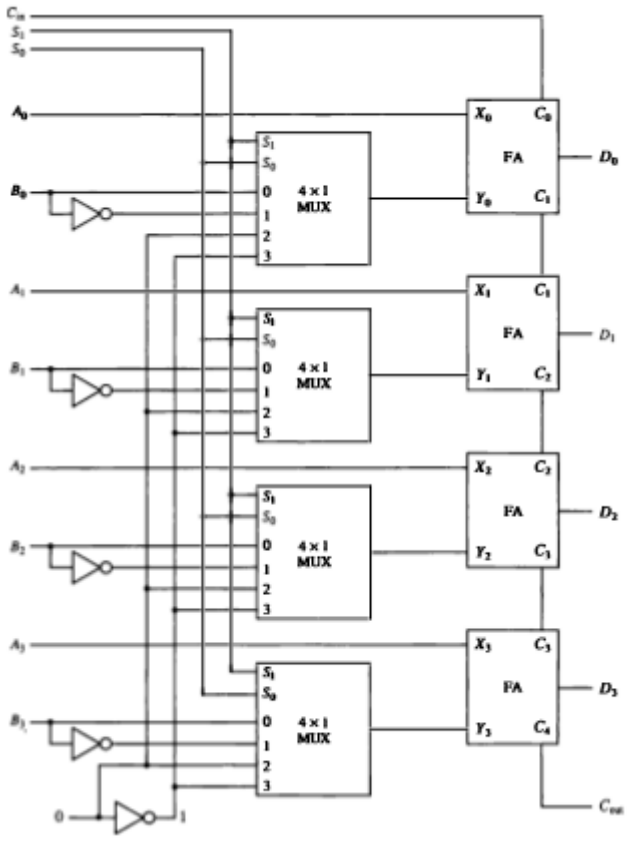
Figure 4-6 4-bit binary adder.

Q.3 Draw block diagram of 4 bit arithmetic circuit and explain

Ans: It has four full-adder circuits that constitute the 4-bit adder and four multiplexers for choosing different operations. There are two 4-bit inputs A and B and a 4-bit output D. The four inputs from A go directly to the X inputs of the binary adder. Each of the four inputs from B are connected to the data inputs of the multiplexers. The multiplexers data inputs also receive the complement of B. The other two data inputs are connected to logic-0 and logic-1. Logic-0 is a fixed voltage value (0 volts for TTL integrated circuits) and the logic-1 signal can be generated through· an inverter whose input is 0. The four multiplexers are controlled by two selection inputs, S1 and 50• The input carry C'" goes to the carry input of the FA in the least significant position. The other carries are connected from one stage to the next. The output of the binary adder is calculated from the following arithmetic sum:

D =A+ Y + C'"

where A is the 4-bit binary number at the X inputs and Y is the 4-bit binary number at the Y inputs of the binary adder. C'" is the input carry, which can be equal to 0 or 1.



Arithmetic circuit function table

Select			Input	Output	
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	Microoperation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

When S1S0 = 00, the value of B is applied to the Y inputs of the adder. IfCin = O, theoutputD =A+ B.IfCin = 1, outputD =A+ B + l. Bothcases perform the add microoperation with or without adding the input carry.

When S1S0 = 01, the complement of B is applied to the Y inputs of the adder. If Cin = 1, then D =A + B + 1. This produces A pius the 2's complement of B, which is equivalent to a subtraction of A-B. When Cm = 0, then D = A + B. This is equivalent to a subtract with borrow, that is, A - B - 1.

When S1S0 = 10, the inputs from B are neglected, and instead, all O's are inserted into the Y inputs. The output becomes D = A + 0 + Cm· This gives D = A when Cm = 0 and D = A + 1 when Cin = 1. In the first case we have a direct transfer from input A to output D. In the second case, the value of A is incremented by 1.

When S1So = 11, all 1' s are inserted into the Y inputs of the adder to produce the decrement operation D = A - 1 when Cm = 0. This is because a number with all 1's is equal to the 2's complement of 1 (the 2's complement of binary 0001 is 1111). Adding a number A to the 2's complement of 1 produces F =A + 2's complement of 1 = A - 1. When Cin = 1, then D =A - 1 + 1 = A, which causes a direct transfer from input A to output D. Note that the microoperation D = A is generated twice, so there are only seven distinct microoperations in the arithmetic circuit.

Q4) Explain selective set, selective component, and selective clear operation.

Ans **Selective-set** sets to 1 the bits in register A where there is a corresponding 1 in register

1010 Content of A before
1100 Content of B (logic operand)
1110 Content of A after

This is done using the logical-OR operation.

Selective-complement complements the bits in register A where there is a corresponding 1 in register B:

1010 Content of A before
1100 Content of B (logic operand)
0110 Content of A after

This is done using the exclusive-OR operation.

Selective-clear clears to 0 the bits in register A where there is a corresponding 1 in register B:

1010 Content of A before
1100 Content of B (logic operand)
0010 Content of A after

This is done using the logical-AND operation and B'.

M. L. V. Govt. Textile Engineering College, Bhilwara Pur Road, Pratap Nagar
Marks 10 Subject (Computer architecture) (V Semester 2017-2018) Time 1 hr

Q.1 Write short note on any one:

a) Priority interrupt
b) DMA

Ans: **Priority Interrupt**: Data transfer between the CPU and the peripherals is initiated by the CPU. But the CPU cannot start the transfer unless the peripheral is ready to communicate with the CPU. When a device is ready to communicate with the CPU, it generates an interrupt signal. A number of input-output devices are attached to the computer and each device is able to generate an interrupt request.

The main job of the interrupt system is to identify the source of the interrupt. There is also a possibility that several devices will request simultaneously for CPU communication. Then, the interrupt system has to decide which device is to be serviced first.

DMA: The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

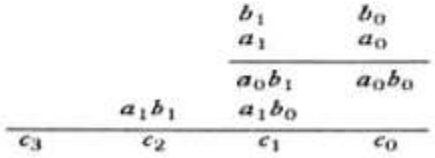
Q2) Explain array multiplier.

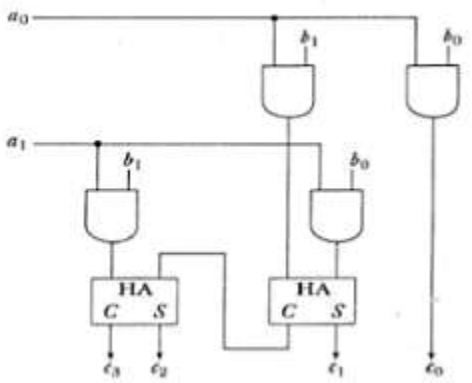
Checking the bits of the multiplier one at a time and forming partial products is a sequential operation that requires a sequence of add and shift microoperations. The multiplication of two binary numbers can be done with one microoperation by means of a combinational circuit that forms the product bits all at once. This is a fast way of multiplying two numbers since all it takes is the time for the signals to propagate through the gates that form the multiplication array. However, an array multiplier requires a large number of gates, and for this reason it was not economical until the development of integrated circuits.

A combinational circuit binary multiplier with more bits can be constructed in a similar fashion. A bit of the multiplier is ANDed with each bit of the multiplicand in as many levels as there are bits in the multiplier. The binary output in each level of AND gates is added in parallel with the partial product of the previous level to form a new partial product. The last level produces the product. For j multiplier bits and k multiplicand bits we need j x k AND gates and (j - 1) k-bit adders to produce a product of j + k bits.

Example:

2-bit by 2-bit array multiplication.
Multiplicand-> b_1 ,b_0
Multiplier->a_1 ,a_0
Product->c_3 ,c_2 ,c_1 ,c_0





Q.3 Explain efficiency, speedup and throughput in pipelining? .

To improve the performance of a CPU we have two options:
1) Improve the hardware by introducing faster circuits.
2) Arrange the hardware such that more than one operation can be performed at the same time.

Since, there is a limit on the speed of hardware and the cost of faster circuits is quite high, we have to adopt the 2nd option.

Pipelining : Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased. Simultaneous execution of more than one instruction takes place in a pipelined processor.

Consider a 'k' segment pipeline with clock cycle time as 'Tp'. Let there be 'n' tasks to be completed in the pipelined processor. Now, the first instruction is going to take 'k' cycles to come out of the pipeline but the other 'n – 1' instructions will take only '1' cycle each, i.e, a total of 'n – 1' cycles. So, time taken to execute 'n' instructions in a pipelined processor:

$$ET_{pipeline} = k + n - 1 \text{ cycles}$$
$$= (k + n - 1)\ Tp$$

In the same case, for a non-pipelined processor, execution time of 'n' instructions will be:

$$ET_{non\text{-}pipeline} = n * k * Tp$$

So, speedup (S) of the pipelined processor over non-pipelined processor, when 'n' tasks are executed on the same processor is:

S = Performance of pipelined processor /
Performance of Non-pipelined processor

As the performance of a processor is inversely proportional to the execution time, we have,

$$S = ET_{non\text{-}pipeline} / ET_{pipeline}$$
$$\Rightarrow S = [n * k * Tp] / [(k + n - 1) * Tp]$$
$$S = [n * k] / [k + n - 1]$$

When the number of tasks 'n' are significantly larger than k, that is, n >> k

$$S = n * k / n$$
$$S = k$$

where ‘k’ are the number of stages in the pipeline.

Also, **Efficiency** = Given speed up / Max speed up = S / S_{max}
We know that, Smax = k

So, **Efficiency** = S / k

Throughput = Number of instructions / Total time to complete the instructions

So, **Throughput** = n / (k + n – 1) * Tp

Q4) Explain arithmetic pipeline with example.

Pipelining is the process of accumulating instruction from the processor through a pipeline. It allows storing and executing instructions in an orderly process. It is also known as **pipeline processing**.

Pipelining is a technique where multiple instructions are overlapped during execution. Pipeline is divided into stages and these stages are connected with one another to form a pipe like structure. Instructions enter from one end and exit from another end.

Pipelining increases the overall instruction throughput.

In pipeline system, each segment consists of an input register followed by a combinational circuit. The register is used to hold data and combinational circuit performs operations on it. The output of combinational circuit is applied to the input register of the next segment.

Arithmetic pipelines are usually found in most of the computers. They are used for floating point operations, multiplication of fixed point numbers etc.

Pipelining is the process of accumulating instruction from the processor through a pipeline. It allows storing and executing instructions in an orderly process. It is also known as **pipeline processing**.

Pipelining is a technique where multiple instructions are overlapped during execution. Pipeline is divided into stages and these stages are connected with one another to form a pipe like structure. Instructions enter from one end and exit from another end.

Pipelining increases the overall instruction throughput.

In pipeline system, each segment consists of an input register followed by a combinational circuit. The register is used to hold data and combinational circuit performs operations on it. The output of combinational circuit is applied to the input register of the next segment.

Pipeline system is like the modern day assembly line setup in factories. For example in a car manufacturing industry, huge assembly lines are setup and at each point, there are robotic arms to perform a certain task, and then the car moves on ahead to the next arm.

Arithmetic Pipeline

Arithmetic pipelines are usually found in most of the computers. They are used for floating point operations, multiplication of fixed point numbers etc. For example: The input to the Floating Point Adder pipeline is:

```
X = A*2^a
Y = B*2^b
```

Here A and B are mantissas (significant digit of floating point numbers), while **a** and **b** are exponents.

The floating point addition and subtraction is done in 4 parts:

1. Compare the exponents.
2. Align the mantissas.
3. Add or subtract mantissas
4. Produce the result.

Registers are used for storing the intermediate results between the above operations