APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

FOURTH SEMESTER BTECH DEGREE EXAMINATION

COMPUTER ORGANISATION & ARCHITECTURE

Previous year question paper -Answer key

1. Differentiate between big endian and little endian byte ordering.

Big-endian and little-endian are terms that describe the order in which a sequence of bytes are stored in computer memory. Big-endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address). Little-endian is an order in which the "little end" (least significant value in the sequence) is stored first. For example, in a big-endian computer, the two bytes required for the hexadecimal number 4F52 would be stored as 4F52 in storage (if 4F is stored at storage address 1000, for example, 52 will be at address 1001). In a little-endian system, it would be stored as 524F (52 at address 1000, 4F at 1001).

Big Endian

In big endian, you store the most significant byte in the smallest address. Here's how it would look:

Address	Value
1000	90
1001	AB
1002	12
1003	CD

Little Endian

In little endian, you store the least significant byte in the smallest address. Here's how it would look:

Address	Value
1000	CD
1001	12
1002	AB
1003	90

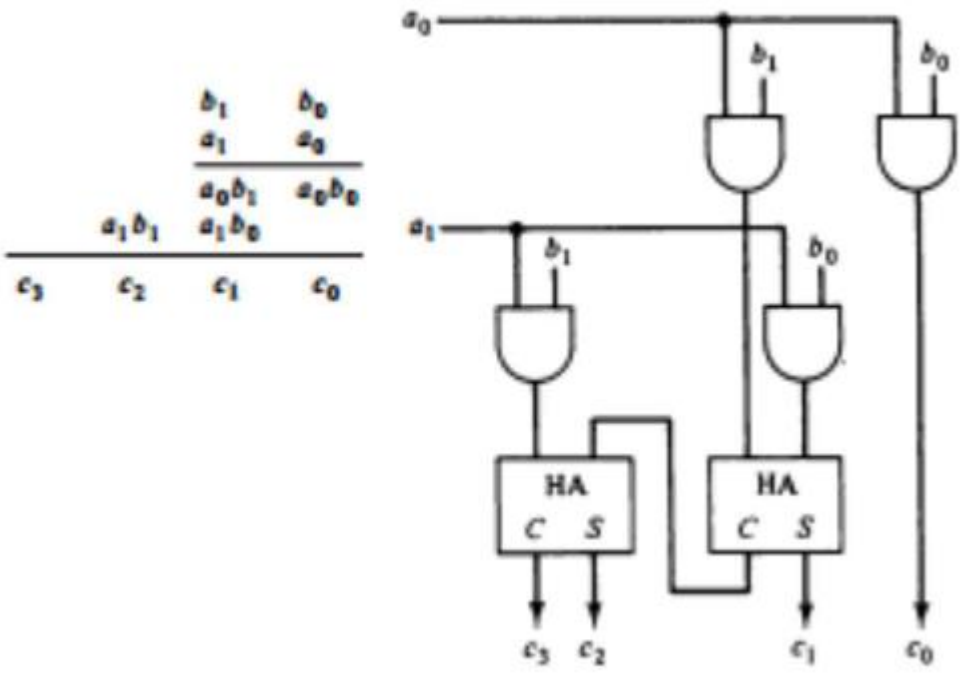
2. Give the control sequence for execution of instruction Add[R3],R1

Step	Action
1	PC_{out} , MAR_{in} , Read, Select4, Add, Z_{in}
2	Z_{out} , PC_{in} , Y_{in} , WMF C
3	MDR_{out} , IR_{in}
4	$R3_{out}$, MAR_{in} , Read
5	$R1_{out}$, Y_{in} , WMF C
6	MDR_{out} , SelectY, Add, Z_{in}
7	Z_{out} , $R1_{in}$, End

3. Design a 2x2 array multiplier

The multiplicand bits are b1 and b0, the multiplier bits are a1 and a0, and the product is c3 c2 c1 c0• The first partial product is formed by multiplying a0 by b1 b0• The multiplication of two bits such as a0 and b0 produces a 1 if both bits are 1; otherwise, it produces a 0. This is identical to an AND operation and can be implemented with an AND gate. As shown in the diagram, the first partial product is formed by means of two AND gates. The second partial product is formed by multiplying a1 by b1 b0 and is shifted one position to the left. The two partial products are added with two half-adder (HA) circuits. Usually, there are more bits in the partial products and it will be necessary to use full-adders to produce the sum. Note that the least significant bit of the product does not have to go through an adder since it is formed by the output of the first AND gate.

Figure 10-9 2-bit by 2-bit array multiplier.



4. a) Describe the different addressing modes. (5)

1. Register Addressing

2. Immediate Addressing

3. PC-Relative Addressing

4. Base Addressing

5. Pseudo-Direct Addressing.

Register Addressing is considered the simplest addressing mode.

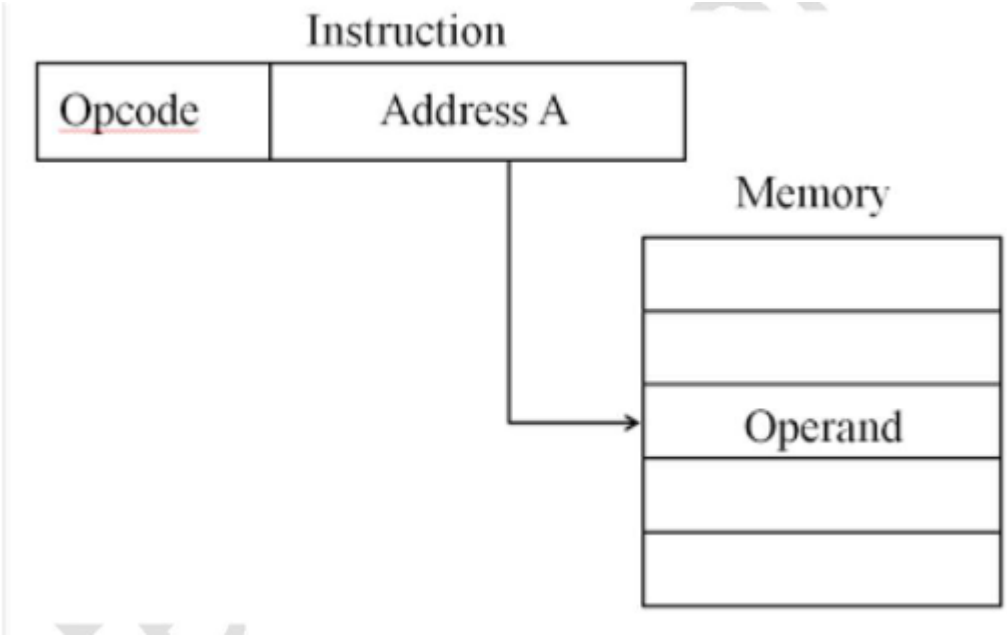
- This is because both operands are in a register. Which allow instructions to be executed much more faster in comparison with other addressing modes because they does not involves with memory access.
- The number of registers is limited since only a few bits are reserved to select a register.
- Register Addressing is a form of direct addressing , this is because we are only interested in the number in the register , rather than using that number as a memory address.
  Here's an example of Register Addressing :

add $s1 , $s2 , $s3 also means that $s1 ←$s2 + $s3

where ; $s1 = rd

$s2 = rs

$s3 = rt



Register Addressing Diagram

That's about it for Register Addressing.

Immediate Addressing is a numeric value embedded in the instruction in the actual operand.

- In immediate addressing , the operand is a constant within the encoded instruction.
- Immediate addressing has the advantage of not requiring an extra memory access to fetch the operand , hence will be executed faster. However , the size of operand is limited to 16 bits.
- The jump instruction format also falls under immediate addressing , where the destination is held in the instruction.

example of Immediate Addressing :

addi $t1 , $zero , 1 means $t1 ← 0 + 7

(add immediate , uses the I-type format)

where ; $t1 = rd

$zero = r1

1 = immediate value

Let's proceed to PC-Relative Addressing.

- PC-Relative Addressing also known as Program Counter Addressing is a data or instruction memory location is specified as an offset relative to the incremented PC.
- PC-relative addressing is usually used in conditional branches. PC refers to special purpose register , Program Counter that stores the address of next instruction to be fetched.
- In PC-relative addressing , the offset value can be an immediate value or an interpreted label value.
- The effective address is the sum of the Program Counter and offset value in the instruction. The effective address determines the branch target.

- PC-relative addressing implements position-independent codes. Only a small offset is adequate for shorter loops.
- Branch instructions can only move 32768 above or below the program counter because the offset is a 16-bit two‟s complement number.
  Another word of saying to explain PC-Relative Addressing :
  The operand address = PC + an offset Implements position-independent codes. A small offset is adequate for short loops.

Example: beqz $t0 , strEnd

where ; $t0 = rs

100 = offset

Thus ; if ($t1 == 0) goto PC + 4 + (4*2)

In this instruction , beqz is a conditional instruction that branches to label in the code if the content of $t0 is equal to zero. If the current address for branch instruction in execution is 0x4000000C , the effective address will be 40000018.

Base Addressing is a data or instruction memory location is specified as a signed offset from a register.

- Base addressing is also known as indirect addressing , where a register act as a pointer to an operand located at the memory location whose address is in the register.
- The register is called base that may point to a structure or some other collection of data and immediate value is loaded at a constant offset from the beginning of the structure. The offset specifies how far the location of the operand data from the memory location pointed by the base.
- The address of the operand is the sum of the offset value and the base value(rs). However, the size of operand is limited to 16 bits because each MIPS instruction fits into a word.
- The offset value is a signed number which is represented in a two's complement format. Therefore , offset value can also be a negative value.

Here's an example for Base Addressing :

Instruction : lw $t1 , 4 ($t2)

where $t1 = rs

$t2 = base (memory address)

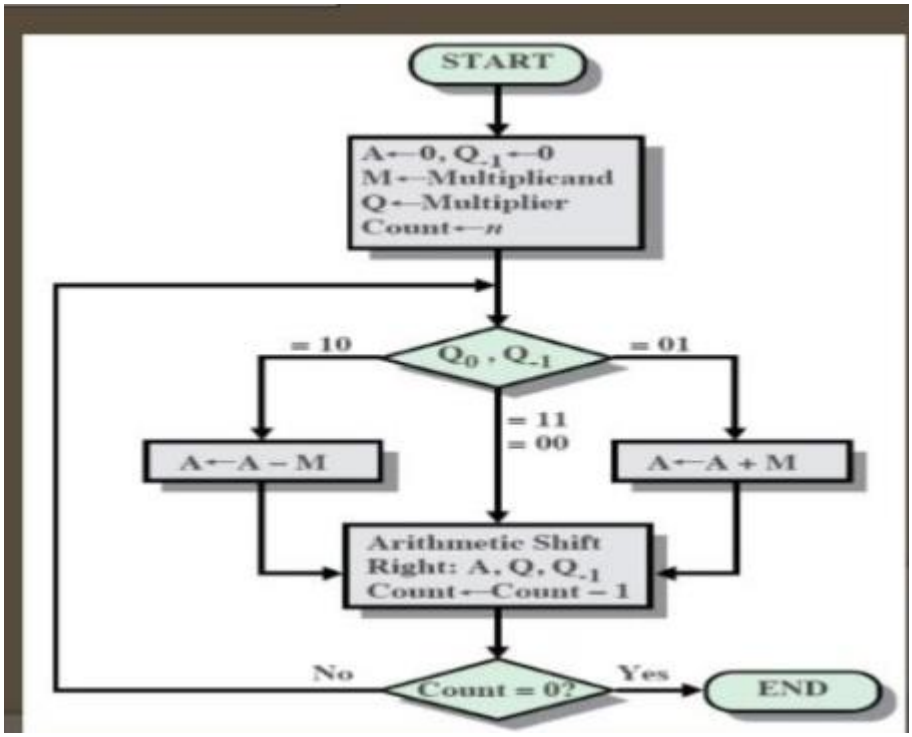
4 = offset value

Thus ; $t1 = Memory [$t2 +4]

In the example above , $t2 pointed to the base of a memory structure. The instruction the load register $t1 with the contents of the memory location four

words onward from the location pointed by register $t2.

b) Give the flow chart for Booth’s Multiplication. (4)



5. Explain restoring method of division with an example.

- Assume ─ that there is an accumulator and MQ register, each of k-bits • MQ 0, (lsb of MQ) bit gives the quotient, which is saved after a subtraction or addition
- Total number of additions or subtractions are k-only and total number of shifts = k plus one addition for restoring remainder if needed
- Assume ─ that X register has (2 k−1) bit for dividend and Y has the k-bitdivisor • Assume ─ a sign-bit S shows the sign

1. Load (upper half k −1 bits of the dividend X) into accumulator k-bit A and load dividend X (lower half bits into the lower k bits at quotient register MQ

• Reset sign S = 0 • Subtract the k bits divisor Y from S-A (1 plus k bits) and assign MQ 0 as per S

2. . If sign of A, S = 0, shift S plus 2 k-bit register pair A-MQ left and subtract the k bits divisor Y from S-A (1 plus k bits); else if sign of A, S = 1, shift S plus 2 k-bit register pair A - MQ left and add the divisor Y into S-A (1 plus k bits)

• Assign MQ 0 as per S

3. Repeat step 2 again till the total number of operations = k.

4. . If at the last step, the sign of A in S = 1, then add Y into S -A to leave the correct remainder into A and also assign MQ 0 as per S, else do nothing.

5. . A has the remainder and MQ has the quotient

Step	S-flag *	First Register for A	Second Register for MQ	Action Taken	Number of operations (instructions)
Start	0	0b0000	0b0000	Clear S, A, MQ	3 for clearing C, A and M
	0	0b0001	0b1110	Load dividend X (lower k bits) in MQ_{k-1} and MQ_0 and dividend higher k–1 bits in A	2 for loading A and MQ
Step 0A	1	1110	1110	Subtract Y from S-A, because S = 0 result in S-A	1
Step 0B	1	1110	1110	$MQ_0 = 0$ as S = 1	1
Step 0C	1	1101	1100	Shift left S-A-M	2
Step 1A	0	0000	1100	Add Y into S-A, because S = 1	1
Step 1B	0	0000	1101	$MQ_0 = 1$ as S = 0	1
Step 1C	0	0001	1010	Shift left S-A-M	2
Step 2A	1	1110	1010	Subtract Y into S-A, because S = 0	1
Step 2B	1	1110	1010	$MQ_0 = 0$ as S = 1	1
Step 2C	1	1101	0100	Shift left S-A-M	2
Step 3A	1	0000	0100	Add Y into S-A, because S = 1	1
Step 3B	0	0000	0101	$MQ_0 = 1$ as S = 0	1
Step 3C	0	0000	1010	Shift C-A-M	2
Last	0	0000	1010	Do not Add Y into S-A, because S = 0 and make no change in MQ_0	1
Answer	0	Remainder = 0,		Quotient Decimal 10	Total 22

6.Write notes on vectored Interrupts.

In a computer, a vectored interrupt is an I/O interrupt that tells the part of the computer that handles I/O interrupts at the hardware level that a request for attention from an I/O device has been received and and also identifies the device that sent the request. Here the device requesting an interrupt may identify itself to the processor by sending a special code over the bus & then the processor start executing the ISR.The code supplied by the processor indicates the starting address of the ISR for the device.The code length ranges from 4 to 8 bits.The location pointed to by the interrupting device is used to store the staring address to ISR. The processor reads this address, called the interrupt vector & loads into PC.

The interrupt vector also includes a new value for the Processor Status Register.

When the processor is ready to receive the interrupt vector code, it activate the interrupt acknowledge (INTA) line

A vectored interrupt is an alternative to a polled interrupt , which requires that the interrupt handler poll or send a signal to each device in turn in order to find out which one sent the interrupt request.

7. Differentiate between synchronous and asynchronous buses.

Synchronous bus:

- Transmitter and receivers are synchronized of clock.

- Data bits are transmitted with synchronization of clock.
- Character is received at constant Rate.
- Data transfer takes place in block.
- Start and stop bit are required to establish communication of each character.
- Used in high – speed transmission.

Asynchronous bus:

- Transmitters and receivers are not synchronized by clock.
- Bit‟s of data are transmitted at constant rate.
- Character may arrive at any rate at receiver.
- Data transfer is character oriented.
- Start and stop bits are required to establish communication of each character.
- Used in low – speed transmission.

8. Briefly explain static memory.

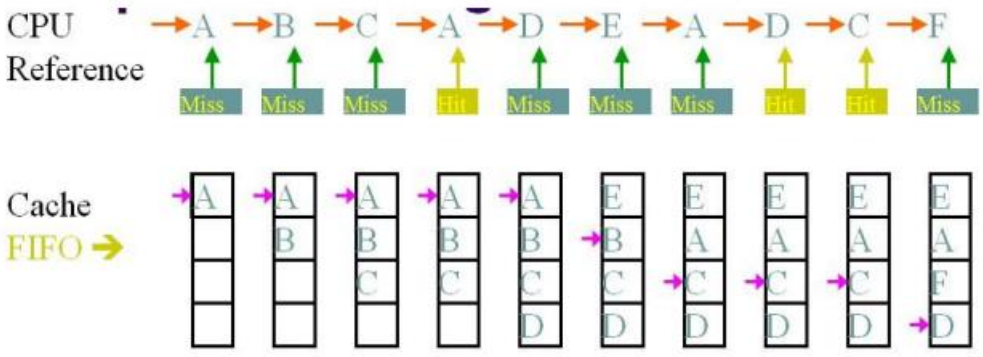
SRAM (static RAM) is random access memory (RAM) that retains data bits in its memory as long as power is being supplied. Unlike dynamic RAM (DRAM), which stores bits in cells consisting of a capacitor and a transistor, SRAM does not have to be periodically refreshed.

Advantages:

- Low power consumption
- Simplicity – a refresh circuit is not needed
- Reliability
- Disadvantages:
- Price
- Capacity

9. Describe the LRU algorithm for cache replacement.

Least recently used (LRU) In the LRU, replace the cache block which is having the less reference with the longest time stamp. In this case also when a hit occurs when the counter value will be set to 0 but when the miss occurs there will be arising of two possibilities in which one possibility is that counter value is set as 0 and in another possibility, the counter value can be incremented as 1.



10. a) Which are the different bus arbitration schemes? (5)

There are two approaches to bus arbitration: Centralized and distributed.

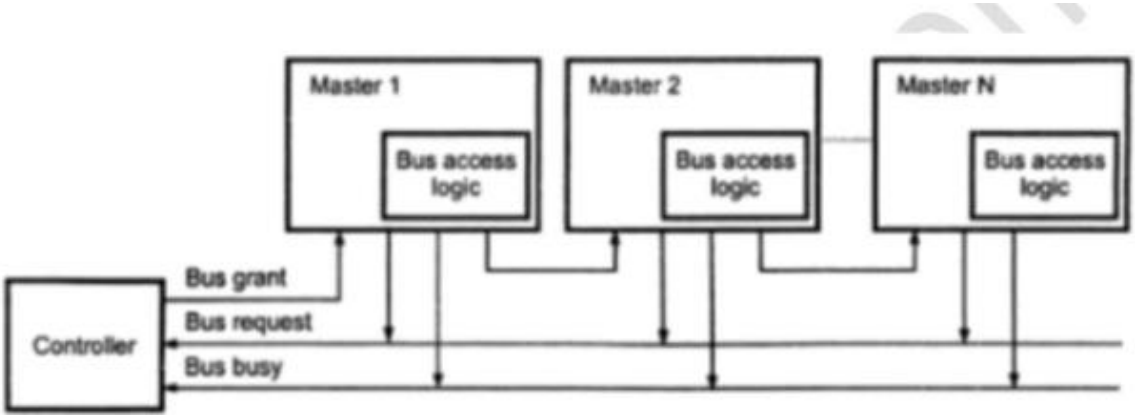
1. Centralized Arbitration

- In centralized bus arbitration, a single bus arbiter performs the required arbitration. The bus arbiter may be the processor or a separate controller connected to the bus.
- There are different arbitration schemes that use the centralized bus arbitration approach. schemes are:

a. Daisy chaining

b. Polling method

o The system connections for Daisy chaining method are shown in fig below.



- It is simple and cheaper method. All masters make use of the same line for bus request.
- In response to the bus request the controller sends a bus grant if the bus is free.
- The bus grant signal serially propagates through each master until it encounters the first one that is requesting access to the bus. This master blocks the propagation of the bus grant signal, activities the busy line and gains control of the bus.
- Therefore any other requesting module will not receive the grant signal and hence cannot get the bus access.

b) Polling method

- In this the controller is used to generate the addresses for the master.
- Number of address line required depends on the number of master connected in the system.
- For example, if there are 8 masters connected in the system, at least three address lines are required.
- In response to the bus request controller generates a sequence of master address. When the requesting master recognizes its address, it activated the busy line ad begins to use the bus.

2. Distributed Arbitration

In distributed arbitration, all devices participate in the selection of the next bus master.

In this scheme each device on the bus is assigned a4-bit identification number.

- The number of devices connected on the bus when one or more devices request for the control of bus, they assert the start-arbitration signal and place their 4-bit ID numbers on arbitration lines, ARB0 through ARB3.
- These four arbitration lines are all open-collector. Therefore, more than one device can place their 4-bit ID number to indicate that they need to control of bus. If one device puts 1 on the

bus line and another device puts 0 on the same bus line, the bus line status will be 0. Device reads the status of all

- lines through inverters buffers so device reads bus status 0as logic 1. Scheme the device having highest ID number has highest priority.
- When two or more devices place their ID number on bus lines then it is necessary to identify the highest ID number on bus lines then it is necessary to identify the highest ID number from the status of bus line. Consider that two devices A and B, having ID number 1 and 6, respectively are requesting the use of the bus.
- Device A puts the bit pattern 0001, and device B puts the bit pattern 0110. With this combination the status of bus-line will be 1000; however because of inverter buffers code seen by both devices is 0111.
- Each device compares the code formed on the arbitration line to its own ID, starting from the most significant bit. If it finds the difference at any bitposition, it disables its drives at that bit position and for all lower-order bits.
- It does so by placing a 0 at the input of their drive. In our example, device detects a different on line ARB2 and hence it disables its drives on line ARB2, ARB1 and ARB0. This causes the code on the arbitration lines tochange to 0110. This means that device B has won the race.
- The decentralized arbitration offers high reliability because operation of the bus is not dependent on any single device.

b) Write notes on flash memory (4)

Flash Memory (sometimes called "Flash RAM") is a type of RAM that, likea ROM, retains its contents when the power supply is removed, but whose contents can be easily erased by applying a short pulse of higher voltage. This is called flash erasure, hence the name. Flash memory is currently both too expensive and too slow to serve as MAIN MEMORY, but is used as removable storage cards for digital cameras and pocket computers.

It is a variation of electrically erasable programmable read-only memory (EEPROM) which, unlike flash memory, is erased and rewritten at the byte level, which is slower than flash memory updating. Flash memory is often used to hold control code such as the basic input/output system (BIOS) in a personal computer. When BIOS needs to be changed (rewritten), the flash memory can be written to in block (rather than byte) sizes, making it easy to update. On the other hand, flash memory is not useful as random access memory (RAM) because RAM needs to be addressable at the byte (not the block) level.

Flash Memory gets its name because the microchip is so organized that a section of memory cells are erased in a single action or "flash." The erasure is caused by tunneling in which electrons pierce through a thin dielectric material to remove an electronic charge from a floating gate associated with each memory cell. Intel offers a form of flash memory that holds two bits (rather than one) in each memorycell, thus doubling the capacity of memory without a corresponding increase in price. Flash memory is used in digital cellular phones, digital cameras, LAN switches, PC Cards for notebook co mputers, digital set-up boxes, embedded controllers, and other devices.

USB, short for Universal Serial Bus, is a standard type of connection for many different kinds of devices.

Generally, USB refers to the types of cables and connectors used to connect these many types of external devices to computers.

The Universal Serial Bus standard has been extremely successful. USB ports and cables are used to connect hardware such as printers, scanners, keyboards, mice, flash drives, external hard drives,

joysticks, cameras, and more to computers of all kinds, including desktops, tablets, laptops, netbooks, etc.

In fact, USB has become so common that you'll find the connection available on nearly any computer-like device such as video game consoles, home audio/visual equipment, and even in many automobiles.

Many portable devices, like smartphones, ebook readers, and small tablets, use USB primarily for charging. USB charging has become so common that it's now easy to find replacement electrical outlets at home improvement stores with USB ports built it, negating the need for a USB power adapter.

11. a) Describe the different types of DRAMS. (5)

DRAM types

The different types of DRAM are used for different applications as a result of their slightly varying properties. The different types are summarised below:

Asynchronous DRAM: Asynchronous DRAM is the basic type of DRAM on which all other types are based. Asynchronous DRAMs have connections for power, address inputs, and bidirectional data lines.

Although this type of DRAM is asynchronous, the system is run by a memory controller which is clocked, and this limits the speed of the system to multiples of the clock rate. Nevertheless the operation of the DRAM itself is not synchronous.

There are various types of asynchronous DRAM within the overall family:

SDRAM: Synchronous DRAM is a type of DRAM that is much faster than previous, conventional forms of RAM and DRAM. It operates in a

synchronous mode, synchronising with the bus within the CPU. RDRAM: This is Rambus DRAM - a type of DRAM that was developed by Rambus Inc, obviously taking its name from the company. It was a competitor to SDRAM and DDR SDRAM, and was able to operate at much faster speeds than previous versions of DRAM.

b) Compare the speed, size and cost of different types of memories

Dynamic RAM (DRAM)

Each memory cell in a DRAM chip holds one bit of data and is composed of a transistor and a capacitor. The transistor functions as a switch that allows the control circuitry on the memory chip to read the capacitor or change its state, while the capacitor is responsible for holding the bit of data in the form of a 1 or 0.

In terms of function, a capacitor is like a container that stores electrons. When this container is full, it designates a 1, while a container empty of electrons designates a 0. However, capacitors have a leakage that causes them to lose this charge, and as a result, the “container” becomes empty after just a few milliseconds.

Thus, in order for a DRAM chip to work, the CPU or memory controller must recharge the capacitors that are filled with electrons (and therefore indicate a

1) before they discharge in order to retain the data. To do this, the memory controller reads the data and then rewrites it. This is called refreshing and occurs thousands of times per second in a DRAM chip. This is also where the “Dynamic” in Dynamic RAM originates, since it refers to the refreshing

necessary to retain the data.

Because of the need to constantly refresh data, which takes time, DRAM is slower.

Static RAM (SRAM)

Static RAM, on the other hand, uses flip-flops, which can be in one of two stable states that the support circuitry can read as either a 1 or a 0. A flip-flop, while requiring six transistors, has the advantage of not needing to be refreshed. The lack of a need to constantly refresh makes SRAM faster than

DRAM; however, because SRAM needs more parts and wiring, an SRAM cell takes up more space on a chip than a DRAM cell does. Thus, SRAM is more expensive, not only because there is less memory per chip (less dense) but also because they are harder to manufacture.

Speed

Because SRAM does not need to refresh, it is typically faster. The average access time of DRAM is about 60 nanoseconds, while SRAM can give access times as low as 10 nanoseconds.

Capacity and Density

Because of its structure, SRAM needs more transistors than DRAM to store a certain amount of data. While a DRAM module only requires one transistor and one capacitor to store every bit of data, SRAM needs 6 transistors. Since the number of transistors in a memory module determines its capacity, for a similar number of transistors, a DRAM module can have up to 6 times more capacity than an SRAM module.

Power Consumption

Typically, an SRAM module consumes less power than a DRAM module. This is because SRAM only requires a small steady current while DRAM requires bursts of power every few milliseconds to refresh. This refresh current is several orders of magnitude greater than the low SRAM standby

current. Thus, SRAM is used in most portable and battery-operated equipment.

However, the power consumption of SRAM does depend on the frequency at which it is accessed. When SRAM is used at a slower pace, it draws nearly negligible power while idled. On the other hand, at higher frequencies, SRAM can consume as much power as DRAM.

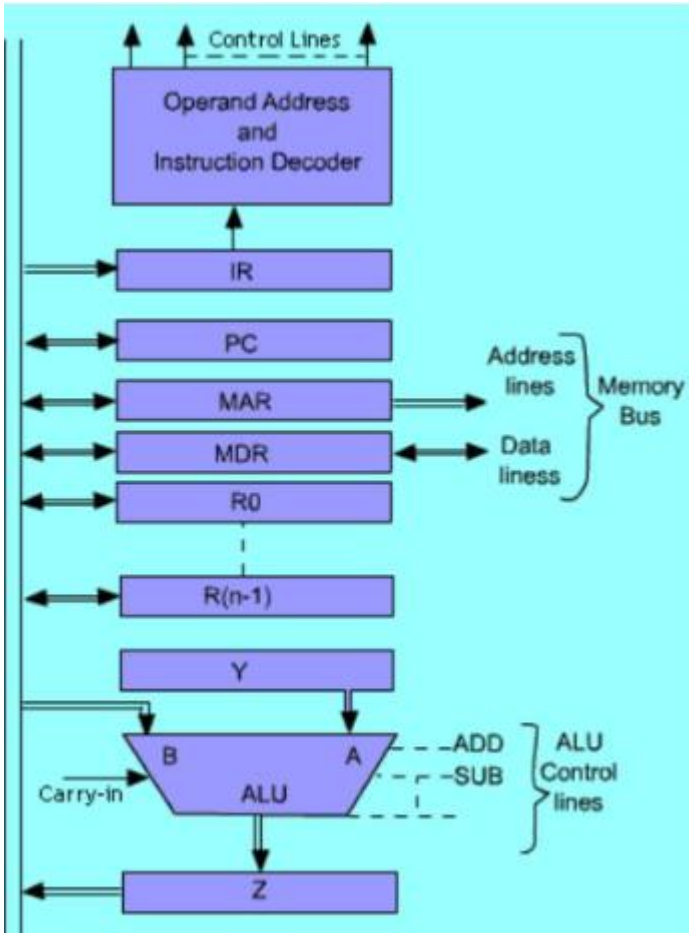
Price

SRAM is much more expensive than DRAM. A gigabyte of SRAM cache costs around $5000, while a gigabyte of DRAM costs $20-$75. Since SRAM uses flip-flops, which can be made of up to 6 transistors, SRAM needs more transistors to store 1 bit than DRAM does, which only uses a single transistor and capacitor. Thus, for the same amount of memory, SRAM requires a higher number of transistors, which increases the production cost.

12. Which are the different methods of processor organization?

Processor Organization

There are several components inside a CPU, namely, ALU, control unit, general purpose register, Instruction registers etc. Now we will see how these components are organized inside CPU. There are several ways to place these components and inteconnect them. One such organization is shown in the Figure



: Single bus organization of the data path inside the CPU

In this case, the arithmatic and logic unit (ALU), and all CPU registers are connected via a single common bus. This bus is internal to CPU and this internal bus is used to transfer the information between different components of the CPU.

This organization is termed as single bus organization, since only one internal bus is used for transferring of information between different components of CPU. We have external bus or buses to CPU also to connect the CPU with the memory module and I/O devices

The number and function of registers R0 to R(n-1) vary considerably from one machine to another. They may be given for general-purpose for the use of the programmer. Alternatively, some of them may be dedicated as special-purpose registers, such as index register or stack pointers.

In this organization, two registers, namely Y and Z are used which are transperant to the user. Programmer can not directly access these two registers. These are used as input and output buffer to the ALU which will be used in ALU operations. They will be used by CPU as temporary storage for some instructions.

For the execution of an instruction, we need to perform an instruction cycle. An instruction cycle consists of two phase,

- Fetch cycle and
- Execution cycle.

Most of the operation of a CPU can be carried out by performing one or more of the following functions in some prespecified sequence:

- Fetch the contents of a given memory location and load them into a CPU register.
- Store a word of data from a CPU register into a given memory location.
- Transfer a word of data from one CPU register to another or to the ALU.
- Perform an arithmatic or logic operation, and store the result in a CPU register.

Now we will examine the way in which each of the above functions is implemented in a computer. Fetching a Word from Memory:

Information is stored in memory location is identified by their address. To fetch a word from memory, the CPU has to specify the address of the memory location where this information is stored and request a Read operation. The information may include both, the data for an operation or the instruction of a program which is available in main memory.

To perform a memory fetch operation, we need to complete the following tasks:

The CPU transfers the address of the required memory location to the Memory Address Register (MAR).

The MAR is connected to the memory address line of the memory bus, hence the address of the required word is transfered to the main memory.

Next, CPU uses the control lines of the memory bus to indicate that a Read operation is initiated. After issuing this request, the CPU waits until it receives an answer from the memory, indicating that the requested operation has been completed.

This is accomplished by another control signal of memory bus known as Memory- Function-Complete (MFC).

The memory set this signal to 1 to indicate that the contents of the specified memory location are available in memory data bus.

As soon as MFC signal is set to 1, the information available in the data bus is loaded into the Memory Data Register (MDR) and this is available for use inside the CPU.

Storing a word into memory

The procedure of writing a word into memory location is similar to that for reading one from memory. The only difference is that the data word to be written is first loaded into the MDR, the write command is issued.

As an example, assumes that the data word to be stored in the memory is in register R1 and that the memory address is in register R2. The memory write operation requires the following sequence:

1. MAR [R2]
2. MDR [R1]
3. Write
4. Wait for MFC

- In this case step 1 and step 2 are independent and so they can be carried out in any order. In fact, step 1 and 2 can be carried out simultaneously, if this is allowed by the architecture, that is, if these two data transfers (memory address and data) do not use the same data path.

In case of both memory read and memory write operation, the total time duration depends on wait for the MFC signal, which depends on the speed of the memory module.

There is a scope to improve the performance of the CPU, if CPU is allowed to perform some other operation while waiting for MFC signal. During the period, CPU can perform some other instructions which do not require the use of MAR and MDR.

Register Transfer Operation

Register transfer operations enable data transfer between various blocks connected to the common bus of CPU. We have several registers inside CPU and it is needed to transfer information from one register another. As for example during memory write operation data from appropriate register must be moved to MDR.

Since the input output lines of all the register are connected to the common internal bus, we need appropriate input output gating. The input and output gates for register R are controlled by the signal Rin and R out respectively.

Thus, when Rin set to 1 the data available in the common bus is loaded into Ri

Similarly when, Ri out is set to 1, the contents of the register Ri are placed on the bus. To transfer data from one register to other register, we need to generate the appropriate register gating signal.

Performing the arithmetic or logic operation:

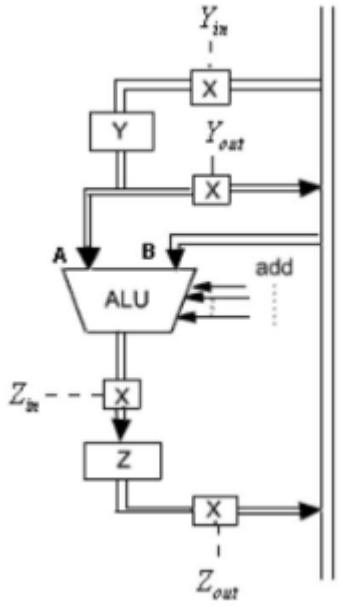
- Generally ALU is used inside CPU to perform arithmetic and logic operation. ALU is a combinational logic circuit which does not have any internal storage.

Therefore, to perform any arithmetic or logic operation (say binary operation) both the input should be made available at the two inputs of the ALU simultaneously.

Once both the inputs are available then appropriate signal is generated to perform the required operation.

We may have to use temporary storage (register) to carry out the operation in ALU

The sequence of operations that have to carried out to perform one ALU operation depends on the organization of the CPU. Consider an organization in which one of the operand of ALU is stored in some temporary register Y and other operand is directly taken from CPU internal bus. The result of the ALU operation is stored in another temporary register Z. This organization is shown in the Figure



Organization for Arithmatic & Logic Operation.

13. Explain the design of a 4bit Arithmetic unit with two selection variables, which performs the basic arithmetic functions.

An Arithmetic and Logic Unit (ALU) is a combinational circuit that performs logic and arithmetic micro-operations on a pair of n-bit operands (ex. A[3:0] and B[3:0]). The operations performed by an ALU are controlled by a set of function-select inputs. In this lab you will design a 4-bit ALU with 3 function-select inputs:

Mode M, Select S1 and S0 inputs. The mode input M selects between a Logic (M=0) and Arithmetic (M=1) operation. The functions performed by the ALU are specified in Table I.

Table 1: Functions of ALU

M = 0 Logic				
S1	S0	C0	FUNCTION	OPERATION (bit wise)
0	0	X	$A_i.B_i$	AND
0	1	X	$A_i + B_i$	OR

1	0	X	$A_i \$ B_i$	XOR
1	1	X	A_i !$ B_i	XNOR
M = 1 Arithmetic				
S1	S0	C0	FUNCTION	OPERATION
0	0	0	A	Transfer A
0	0	1	A + 1	Increment A by 1
0	1	0	A + B	Add A and B
0	1	1	A + B + 1	Increment the sum of A and B by 1
1	0	0	A + B'	A plus one's complement of B
1	0	1	A - B	Subtract B from A (i.e. B' + A + 1)
1	1	0	A' + B	B plus one's complement of A
1	1	1	B - A	B minus A (or A' + B + 1)

Figure 1: Block diagram of the 4-bit ALU.

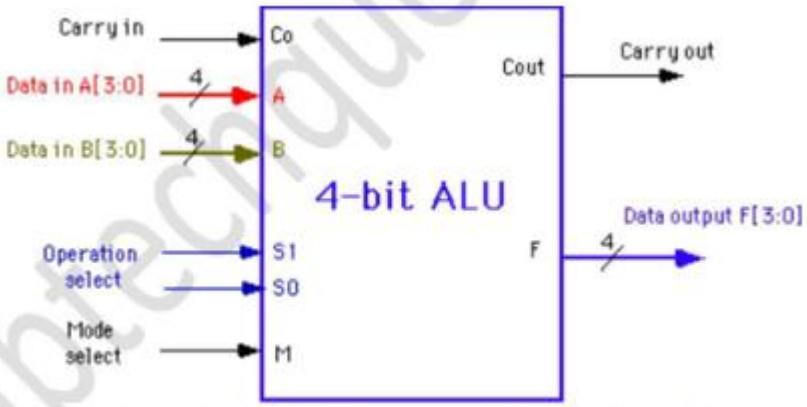


Figure 1: Block diagram of the 4-bit ALU.

When doing arithmetic, we need to decide how to represent negative numbers. As is commonly done in digital systems, negative numbers are represented in twos complement. This has a number of advantages over the sign and magnitude representation such as easy addition or subtraction of mixed positive and negative numbers. Also, the number zero has a unique representation in twos complement.

14. a) Explain the design of status register. (5)

The STATUS register is of most importance to programming the PIC, it contains the arithmetic status of the ALU (Arithmetic Logic Unit), the RESET status and the bank select bit for data memory. As

with any register, the STATUS register can be the destination for any instruction. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to device logic. Furthermore, the TO and PD bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended. For example, CLRF STATUS will clear the upper-three bits and set the Z bit. This leaves the STATUS register as 000u u1uu (where u = unchanged).

The first three bits (STATUS<0> to STATUS<2>) are the carry (C), digit carry (DC) and zero (Z) flags of the ALU respectively. The values of these bits change depending on the results of arithmetic or logical operations performed during program execution. Bits 3 and 4 are the power down PD and watchdog timer timeout TO bits respectively and bits 5and6(RP0 and RP1) are the bank selection bits.

bit 7:

IRP: Register Bank Select bit (used for indirect addressing)

0 = Bank 0, 1 (00h - FFh)

1 = Bank 2, 3 (100h - 1FFh)

The IRP bit is not used by the PIC16F8X. IRP should be maintained clear.

bit 6-5:

RP1:RP0: Register Bank Select bits (used for direct addressing)

00 = Bank 0 (00h - 7Fh)

01 = Bank 1 (80h - FFh)

10 = Bank 2 (100h - 17Fh)

11 = Bank 3 (180h - 1FFh)

Each bank is 128 bytes. Only bit RP0 is used by the PIC16F8X. RP1 should be

maintained clear.

bit 4:

TO: Time-out bit

1 = After power-up, CLRWDT instruction, or SLEEP instruction

0 = A WDT time-out occurred

bit 3:

PD: Power-down bit

1 = After power-up or by the CLRWDT instruction

0 = By execution of the SLEEP instruction

bit 2:

Z: Zero bit

1 = The result of an arithmetic or logic operation is zero

0 = The result of an arithmetic or logic operation is not zero

bit (for ADDWF and ADDLW instructions) (For borrow the polarity is reversed)

bit 1:

DC: Digit carry/borrow

1 = A carry-out from the 4th low order bit of the result occurred

0 = No carry-out from the 4th low order bit of the result

bit (for ADDWF and ADDLW instructions)

bit 0:

C: Carry/borrow

1 = A carry-out from the most significant bit of the result occurred

0 = No carry-out from the most significant bit of the result occurred

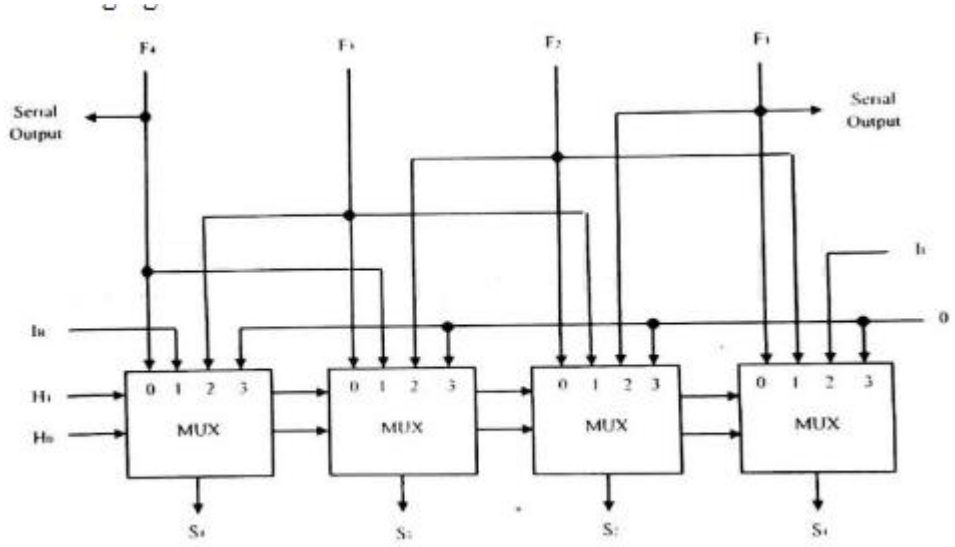
Note: For borrow the second operand the polarity is reversed. A subtraction is

executed by adding the two‟s complement of. For rotate (RRF, RLF) instructions,

this bit is loaded with either the high or low order bit of the source register.

b) Give the design of a 4 bit shifter. (5)

4-bit Serial-in to Parallel-out Shift Register



The operation is as follows. Lets assume that all the flip-flops (FFA to FFD) have just been RESET (CLEAR input) and that all the outputs Q to Q are at logicA D level “0” ie, no parallel data output.

If a logic “1” is connected to the DATA input pin of FFA then on the first clock pulse the output of FFA and therefore the resulting QA will be set HIGH to logic “1” with all the other outputs still remaining LOW at logic “0”. Assume now that the DATA input pin of FFA has returned LOW again to logic “0” giving us one data pulse or 0-1-0.

The second clock pulse will change the output of FFA to logic “0” and the output of FFB and QB HIGH to logic “1” as its input D has the logic “1” level on it from QA. The logic “1” has now moved or been “shifted” one place along the register to the right as it is now at QA.

When the third clock pulse arrives this logic “1” value moves to the output of FFC (QC) and so on until the arrival of the fifth clock pulse which sets all the outputs Q to Q back again to logic level “0” because the input to FFA hasA D remained constant at logic level “0”.

The effect of each clock pulse is to shift the data contents of each stage one place to the right, and this is shown in the following table until the complete data value of 0-0-0-1 is stored in the register. This data value can now be read directly fromthe outputs of Q to Q .A D

15. Explain the design of micro program sequencer with an example.?

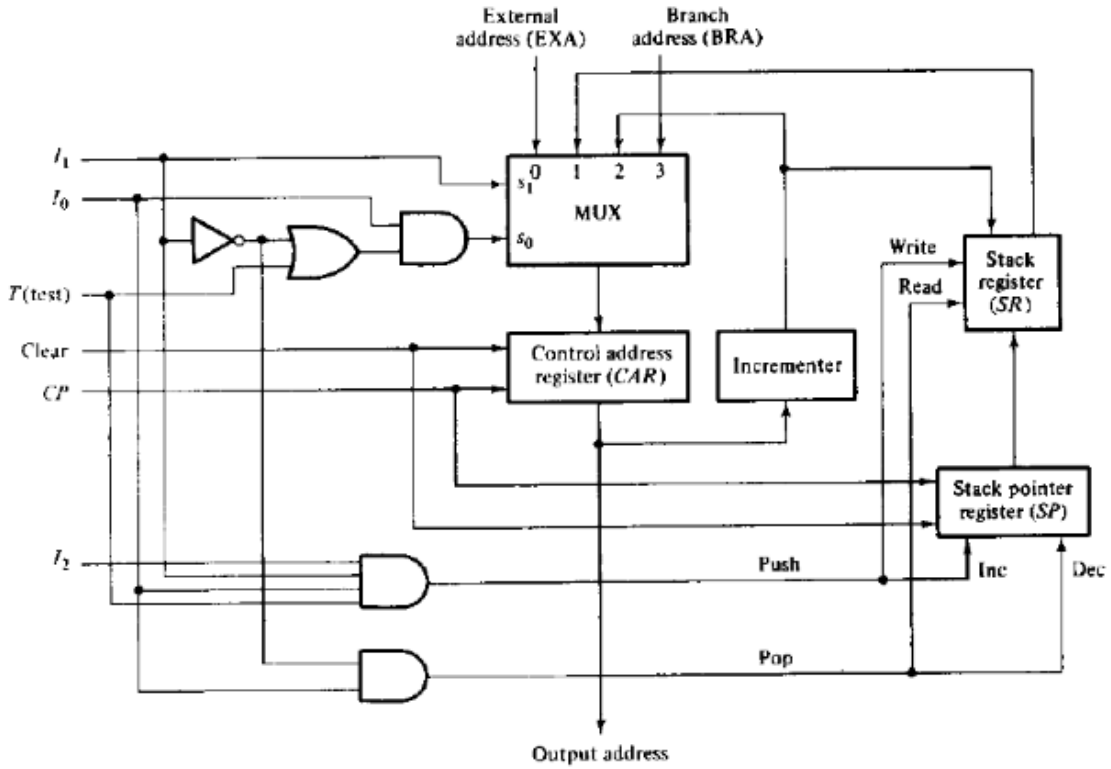
Ans . A micro-program sequencer works in a way to generate these control signals from the microprogram by transitioning from one state to another in every clock cycle. A state is defined by the micro-instruction that has to be run in that clock cycle.It has two main functions

1. Control Function The micro-operations that need to be executed to perform a certain microinstruction are to be defined and be known. The micro-operation(s) are dependent on parameters like selected destination, operand etc.

2. Sequencing Function The address of next micro-instruction to be executed is generated while

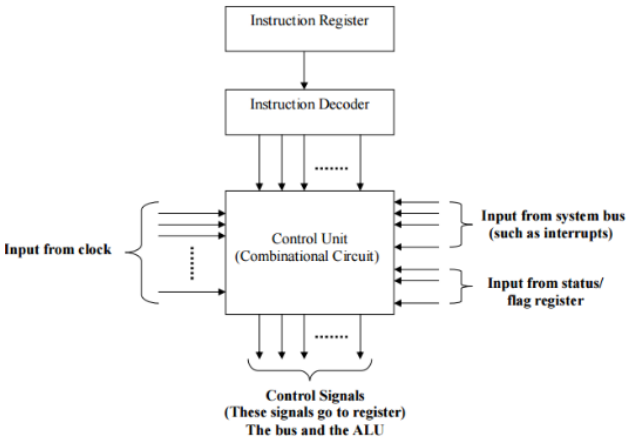
controlling test conditions etc. Thus, to summarize, to execute an instruction, the microprogram sequencer executes a micro-instruction in every clock cycle and determines which micro-instruction (state) to run next. It can be thought in terms of a state diagram. The Instruction Register loads the opcode into the decoder which then translates

the opcode into a control memory address. The control address register contains the address of the next micro-instruction to be read. The micro-instructions are stored in the microprogram memory (also called control memory, can be used interchangeably). The address from the control address register is used to read from this microprogram memory. When a micro-instruction is read from the microprogram memory, it is transferred to the control buffer register. This register activates the control signals. Thus, reading a micro-instruction from the microprogram memory has the effect of executing that micro-instruction. The sequencing logic loads the control address register and activates the read signal. This read signal loads the next micro- instruction from the Instruction Register completing a cycle.



16. Explain the procedure for designing a hardwired control?

Ans. For each instruction, the control unit causes the CPU to execute a sequence of steps correctly. In reality, there must be control signals to assert lines on various digital components to make things happen. For example, when we perform an Add instruction in assembly language, we assume the addition takes place because the control signals for the ALU are set to "add" and the result is put into the AC. The ALU has various control lines that determine which operation to perform. The question we need to answer is, "How do these control lines actually become asserted?" We can take one of two approaches to ensure control lines are set properly. The first approach is to physically connect all of the control lines to the actual machine instructions. The instructions are divided up into fields, and different bits in the instruction are combined through various digital logic components to drive the control lines. This is called hardwired control, and is illustrated in figure (1). The control unit is implemented using hardware (for example: NAND gates, flip-flops, and counters).We need a special digital circuit that uses , as inputs, the bits from the Opcode field in our instructions, bits from the flag (or status) register, signals from the bus, and signals from the clock. It should produce, as outputs, the control signals to drive the various components in the computer. The advantage of hardwired control is that is very fast. The disadvantage is that the instruction set and the control logic are directly tied together by special circuits that are complex and difficult to design or modify.



17 a) Explain the different methods of control organization. (5)

Ans Once a control sequence has been established, the sequential system that implements the control operations must be designed. Since the control is a sequential circuit, it can be designed by a sequential logic procedure. Disadvantages of sequential control logic are

• Large number of states

• Excessive number of flip-flops and gates

• Design methods uses state and excitation tables but in practice they are cumbersome

Goal of control logic design should be development of a circuit that implements the desired control sequence in a logical andstraightforward manner. Designers used specialized methods for control logic design which is considered as the extension of the classical sequential logic method combined with register transfer method.

We consider four methods of control organization

• One flip-flop per state methods

• Sequence register and decoder method

• PLA control

• Micro-program control

The first two methods result in a circuit that must use SSI and MSI circuits for the implementation.A control unit implemented with SSI and MSI devices is said to be a hard-wired control.If any alterations or modifications are needed, the circuits must be rewired to fulfill the new requirements.

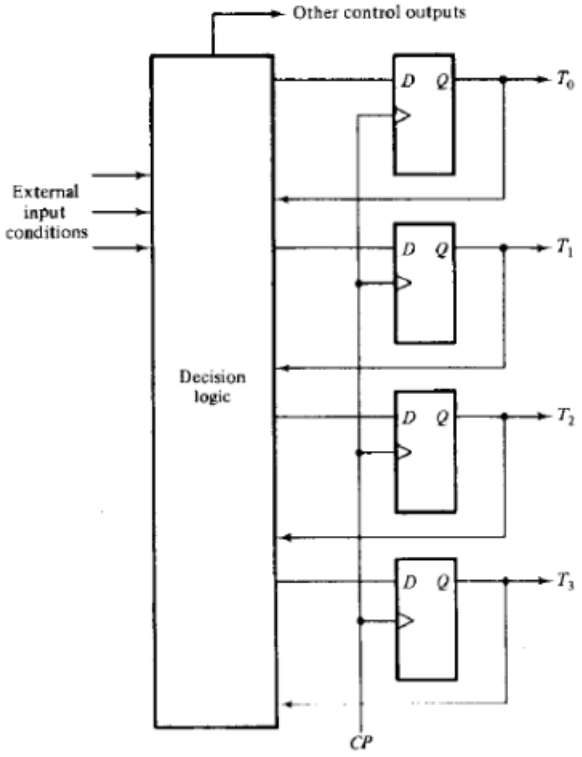
The PLA or micro-program control which uses an LSI device such as a programmable logic array or a read-only memory. Any alterations or modifications in a micro-program control can be easily achieved without wiring changes by removing the ROM from its socket and inserting another ROM programmed to fulfill the new specifications.

One flip-flop per state methods

This method uses one flip-flop per state in the control sequential circuit. Only one flip-flop is set at any particular time: all others are cleared. A single bit is made to propagate from one flipflop to the other under the control of decision logic. In such an array, each flip-flop represents a state and is activated only when the control bit is transferred to it.

In this method, maximum numbers of flipflops were used. Example: A sequential circuit with 12 states requires a minimum of four flipflops because 23< 12< 24 . Control circuit uses 12 flip-flops, one foreach state

The advantage of this method is the simplicity with which it can be designed. This type of controller can be designed by inspection from the state diagram that describes the control sequence. This also offers other advantages like savings in design effort, an increase in operational simplicity, and a potential decrease in the combinational circuits required to implement the complete sequential circuit. The disadvantage is that this method would increase system cost since more flip-flops are used. Figure shows the configuration of a four-state sequential control logic that uses four Dtype flip-flops: one flip-flop per state Ti, i = 0, 1, 2, 3.



At any given time interval between 2 clock pulses, only one flip-flop is equal to 1; all others are 0. The transition from the present state to next state is a function of the present Ti that is a 1 and certain input conditions. The next state is manifested when the previous flip-flop is cleared and a new one is set. Each of the flip-flops is connected to the data processing section of the digital system to initiate certain micro-operations. The control outputs are a function of the T's and external inputs. These outputs may also initiate micro-operations If control circuit does not need external inputs for its sequencing, the circuit reduces to straight shift register with a single bit shifted from one position to the next. If control sequence must repeated over and over again the control reduces to ring counter. Ring counter is a shift register with the output of last flip-flop connected to the input of the first flip-flop.In a ring counter single bit continuously shifts from one position to the next in a circular manner. For this reason, this method is also called ring counter controller

Sequence Register and Decoder Method

This method uses a register to sequence the control states. The register is decoded to provide one output for each state. For n flip-flops in the sequence register, the circuit will have 2" states and the decoder will have 2" outputs. For example, a 4-bit register can be in any one of 16 states. A 4 x 16 decoder will have 16 outputs, one for each state of the register. Both the sequence register and decoder are MS1 devices.

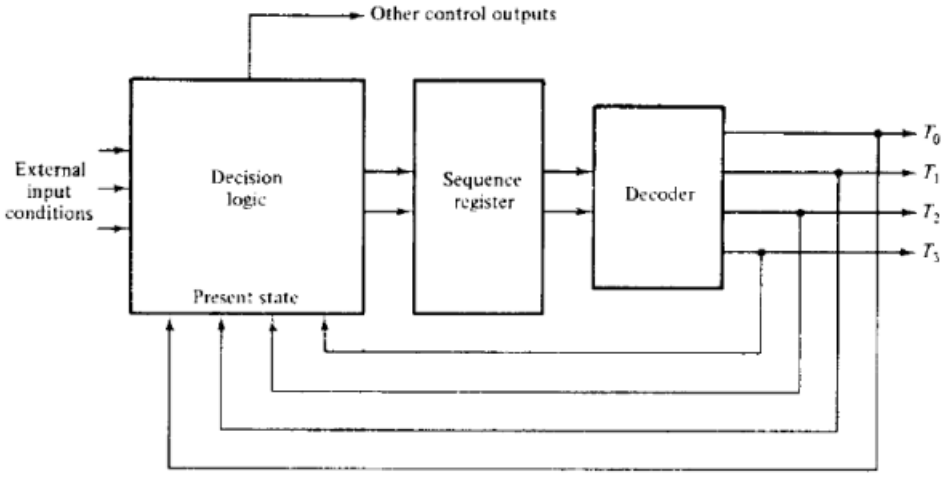
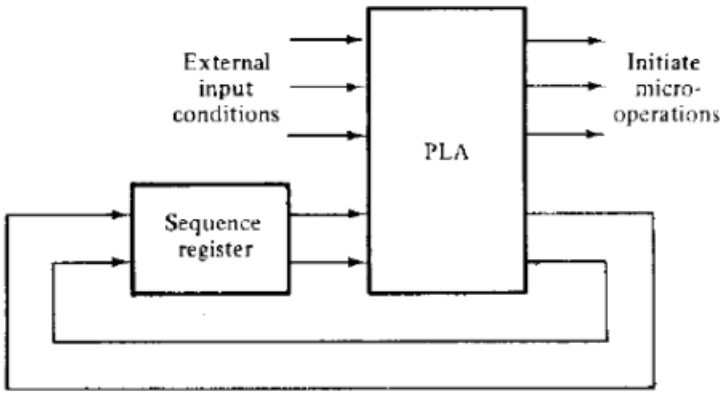


Figure shows the configuration of a four-state sequential control logic. The sequence register has two flip-flops and the decoder establishes separate outputs for each state in the register. The transition to the next state in the sequence register is a function of the present state and the external input conditions.

PLA control

The external sequence register establishes the present state of the control circuit. The PLA outputs determine which micro-operations should be initiated depending on the external input conditions and the present state of the sequence register. At the same time other PLA outputs determine the next state of the sequence register.



The sequence register is external to the PLA if the unit implements only combinational circuits. Some PLAs include not only gates but also flip-flops within the unit. This implement a sequential circuit by specifying the links that must be connected to the flip-flops in manner that the gate links are specified

Micro-program Control

The purpose of control unit is to initiate a series of sequential steps of micro-operations. At any given time certain operations are to be initiated while all others remain idle. The control variable at any given time can be represented by a string of 1's and 0's called control word. The control words can be programmed to initiate the various components in the system in an organized manner. A control unit whose control variables are stored in a memory called a micro-programmed control unit. Each control word of memory is called Microinstruction and Sequence of microinstructions is called Micro-program. Control memory is usually ROM since an alteration of micro-program is seldom needed. The use of micro-program involves placing all control variables in words of the ROM for use by the control unit through successive read operations.The content of the word in the ROM at a given address specifies the micro-operations for the system.

Dynamic micro-programming permits a micro-program to be loaded initially from the computer console or from an auxiliary memory such as magnetic disk. Writable control memory(WCM) can be used for writing but used mostly for reading. A ROM, PLA or WCM when used in a control unit is referred as a control memory Control memory address register specifies the control word read from control memory. The ROM operates as a combinational circuit with address value as the input and the corresponding word as the output. The content of the specified word remains on the output wires as long as the address value remains in the address register. If the address registers changes while the ROM word is still in use then the word out of the ROM should be transferred to a buffer register. If the change in address and ROM word can occur simultaneously no buffer register is needed. The word read from memory represents a microinstruction. The microinstruction specifies one or more micro-operations for the components of the system. Once these operations are executed, the control unit must determine its next address. The location of the next microinstruction may be next one in the sequence or it may locate somewhere else in the control memory. Some bits of the microinstruction to control the generation of the address for the next microinstruction. The next address may be function of external input conditions. The next address is computed in the next address generator circuit and then transferred into the control address register to read the next microinstruction.