II - ASSIGNMENT

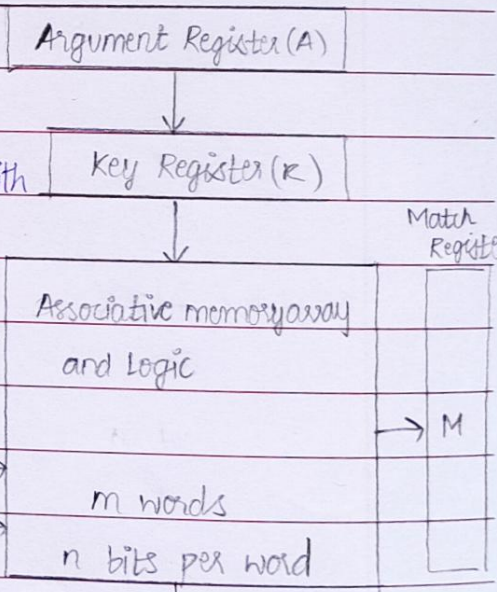
(Start Writing From Here)

① Explain about Associative Memory.

ASSOCIATIVE MEMORY

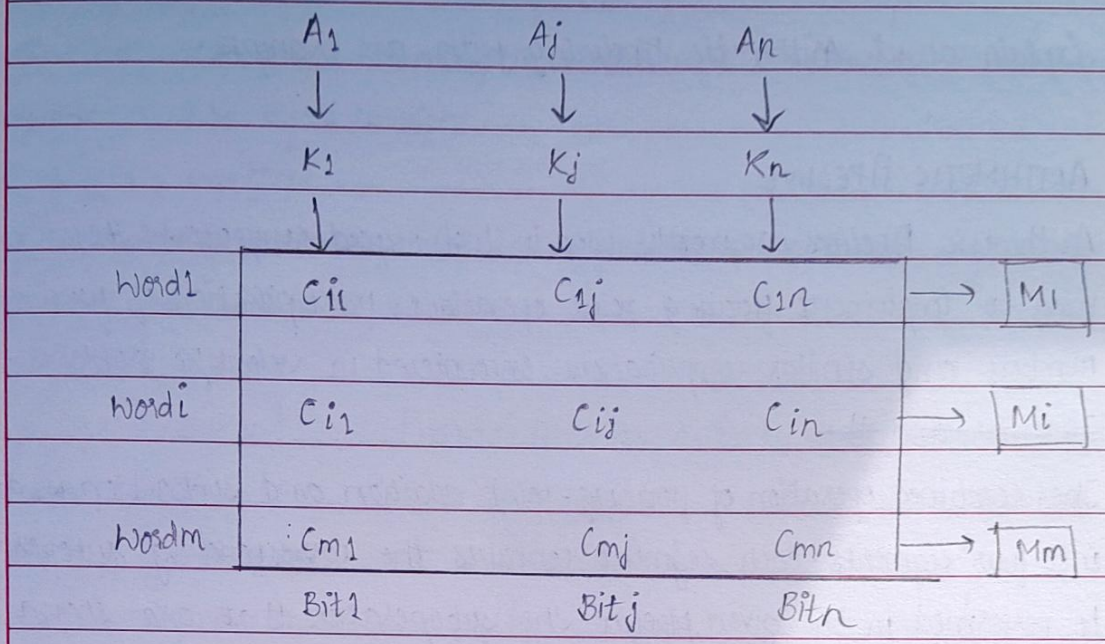
An associative memory can be considered as a memory unit whose stored data can be identified for access by the content of data itself rather than by an address or a memory location.

Associative memory is often refered to as Content Addressable Memory (CAM). When a write operation is performed on an associative memory, no address or memory location is given to word - the memory itself finds an empty location to store. When the word is to be red from associative memory, the content of word or part of word is specified. The words matching the specified content are located and are marked for reading.



BLOCK REPRESENTATION OF ASSOCIATIVE MEMORY

From the diagram, we can say an associative memory consists of array and logic for 'm' words with 'n' bits per word. The functional registers like the argument register A & key register K each have n bits, one for each bit of a word. The match register M consists of m bits, one for each memory word. The words which are kept in the memory are compared in parallel with the content of the argument register.



The cells present inside the memory array are marked by the letter C with two subscripts. The first subscript gives the word number and the second specifies the bit position in the word. For instance, the cell Cij is the cell for bit j in word i.

A bit Aj in the argument register is compared with all the bits in column j of the array provided that Kj = 1. This process is done for all columns j = 1, 2, 3 - - - n.

If a match occurs between all the unmasked bits of the argument and the bits in word i, the corresponding bit Mi in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match. Mi is cleared to 0.

(2) Explain about Arithmetic Pipelining with an example.

ARITHMETIC PIPELINE

Arithmetic Pipelines are mostly used in high-speed computers. They are used to implement floating point operations, multiplication of fixed-point numbers and similar computations encountered in scientific problems.

~~To understand the~~

The combined operation of floating-point addition and subtraction is divided into four segments. Each segment contains the corresponding suboperation to be performed in the given pipeline. The suboperations that are shown in the four segments are:-

(i) Compare the exponents by subtraction
(ii) Align the mantissas
(iii) Add or subtract the mantissas
(iv) Normalize the result

To understand the concepts of arithmetic pipeline in a more convinient way let us consider an example of a pipeline unit for floating-point addition and subtraction.

The inputs to the floating point adder pipeline are two normalized floating-point binary numbers defined as:

$$X = A * 2^a = 0.9504 * 10^3$$

$$Y = B * 2^b = 0.8200 * 10^2$$

where A & B are two fractions that represent mantissa & a, b are the exponents.

1. Compare exponents by subtraction

The exponents are compared by subtracting them to determine their difference.

The larger exponent is chosen as exponent of result. The difference of exponents i.e., 3 - 2 = 1 determines how many times the mantissa associated with smaller exponent must be shifted to right.

2. Align the mantissa

The mantissa of smaller exponent is shifted according to the difference of exponents determined in segment one.

$X = 0.9504 * 10^3$

$Y = 0.08200 * 10^3$

3. Add Mantissas

The two mantissas are added in segment three.

$Z = X + Y$

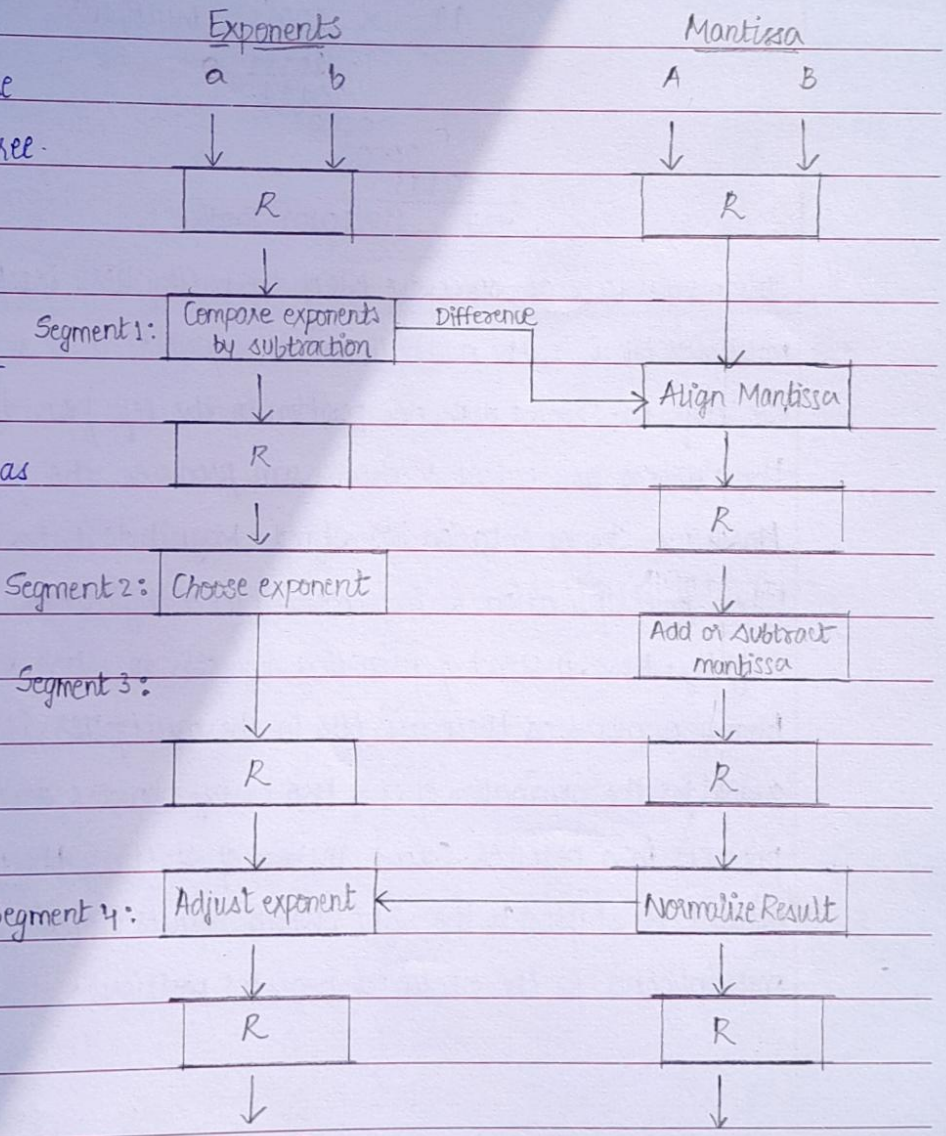
$= 1.0324 * 10^3$

4. Normalize the result

After normalization, the result is written as

$Z = 0.1324 * 10^4$

PIPELINE ORGANIZATION FOR FLOATING POINT ADDITION & SUBTRACTION



CMR GROUP OF INSTITUTIONS
EXPLORE TO INVENT

(5) With an example, explain multiplication algorithm.

MULTIPLICATION ALGORITHM

Multiplication of two fixed-point binary numbers in signed magnitude representation is done with paper & pencil with a process of successive shift and add operations. This process is best illustrated with a numerical example:—

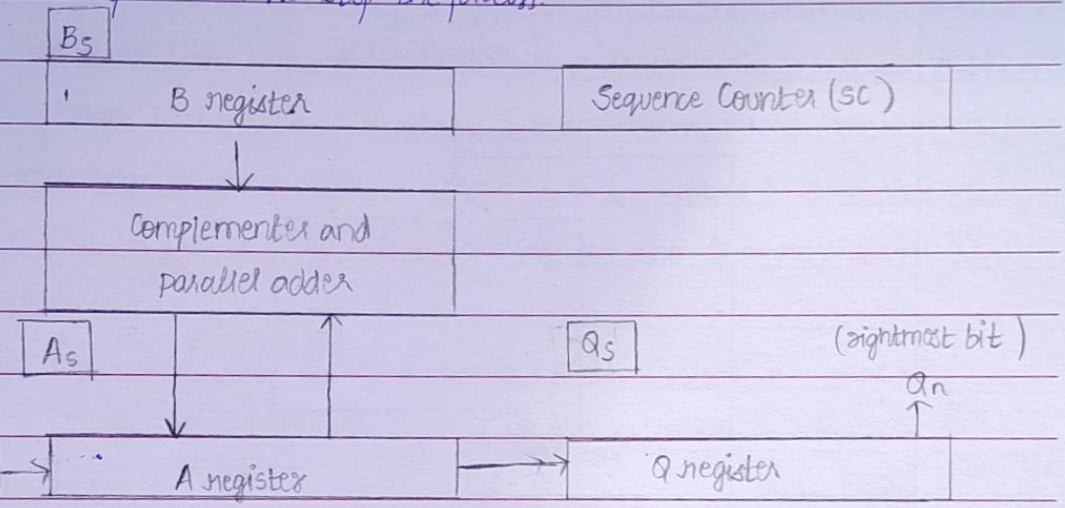
```
      23          10111    Multiplicand
      19      x   10011    Multiplier
    ---------------------
                  10111
                 10111
                00000
               00000
              10111
    ---------------------
     437       110110101   Product
```

This process looks at successive bits of the multiplier, least significant bits first. If the multiplier bit is 1, the multiplicand is copied as it is; otherwise we copy zeros. Now we shift no.s copied down one position to the left from the previous numbers. Finally, the numbers are added & their sum produces the product.

Hardware Implementation For Signed-Magnitude Data

When multiplication is implemented in a digital computer, we change the process slightly. Here, instead of providing registers to store & add simultaneously as many binary numbers as there are bits in the multiplier, it is convinient to provide an adder for the summation of only two binary numbers and accumulate the partial products in a register. Second, instead of shifting the multiplicand to left, the partial product is shifted to the right, which results in leaving the partial product and the multiplicand in the required register positions.

The hardware for the multiplication consists of the equipment as given below. The multiplier is stored in the register and its sign in Q_S. The sequence counter SC is initially set bits in the multiplier. After forming each partial product the counter is decremented. When the content of the counter reaches zero the product is complete and we stop the process.

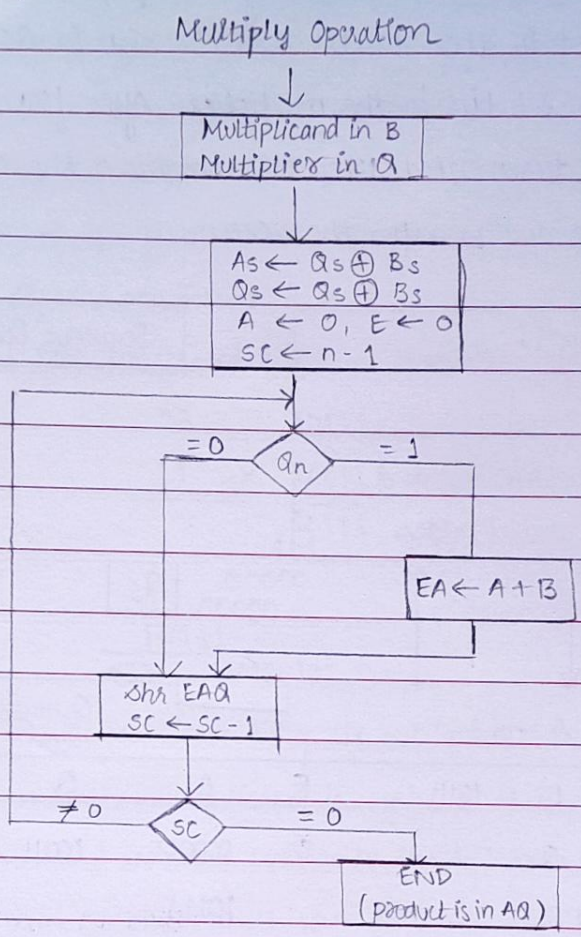


Example:- Multiplicand B = 10111

	E	A	Q	SC
Multiplier in Q	0	00000	10011	101 (5)
Q_n = 1; Add B		10111		
First partial product	0	10111		
shift right EAQ	0	01011	11001	100 (4)
Q_n = 1; Add B		10111		
Second partial product	1	00010		
shift right EAQ	0	10001	01100	011 (3)
Q_n = 0; shift right EAQ	0	01000	10110	010 (2)
Q_n = 0; shift right EAQ	0	00100	01011	001 (1)
Q_n = 1; add B		10111		
Fifth partial product	0	11011		
shift right EAQ	0	01101	10101	000 (0)

Final product in AQ = 0110110101

Hardware Algorithm



(4) Explain Asynchronous data transfer and communication interface.

ASYNCHRONOUS DATA TRANSFER

The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator. If the registers in the interface share a common clock with CPU registers the transfer is synchronous. In most cases, the internal timing in each unit is independent from the others in that each uses its own private clock for internal registers, which is said to be Asynchronous.

Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted. One way is by strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur. Another method used is to accompany each data item being transferred with a control signal that indicates the presence of data in bus. The unit receiving data responds with another control signal to acknowledge receipt of data. This agreement is referred to as Handshaking. Generally, we considers transmitting unit as the source and the receiving unit as the destination.

Two Types of Asynchronous Data Transmission :- Strobe Control
Handshaking

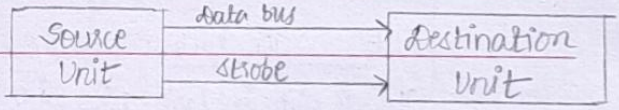
I. STROBE CONTROL

This method employs a single control line to time each transfer. The strobe may be activated by either source or the destination unit.

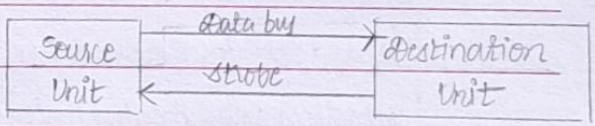
(i) Source - Initiated Transfer

• The data bus carries the binary information from source to destination unit using multiple lines to transfer an entire byte/word. The strobe is a single line that tells the destination unit when a valid data is available in the bus.

(a) BLOCK DIAGRAM



Source initiated

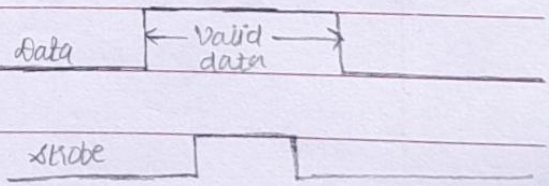


Destination initiated

(ii) Destination - Initiated Transfer

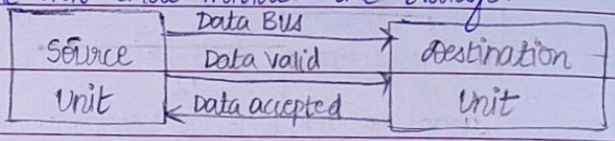
In this case, destination unit activates the strobe pulse informing the source to provide data. The source places data on data bus.

(b) TIMING DIAGRAM

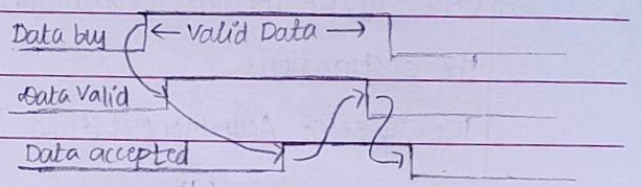


II. HANDSHAKING

- The disadvantage of the strobe method is that source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus. Same for when destination initiates transfer.
- The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer.
- One control line is in the same direction as the data flow in the bus from source to destination.

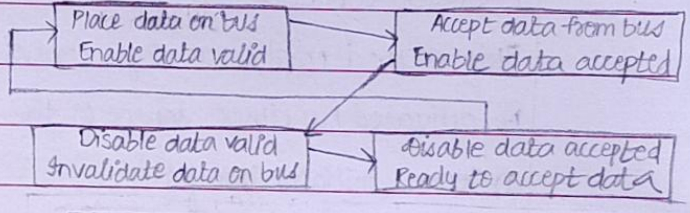


- Other control line is in other direction from destination to the source.



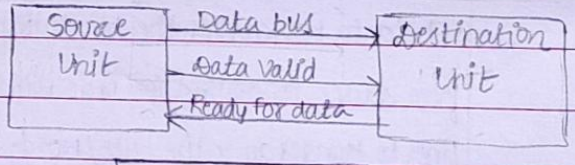
(i) Source-initiated transfer

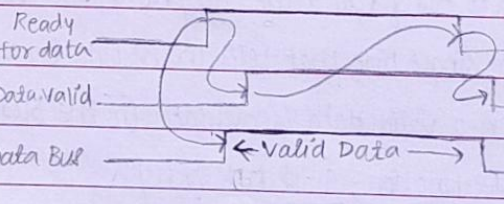
The two handshaking lines are: data valid by source unit & data accepted by destination unit. The sequence of states is shown by diagram.

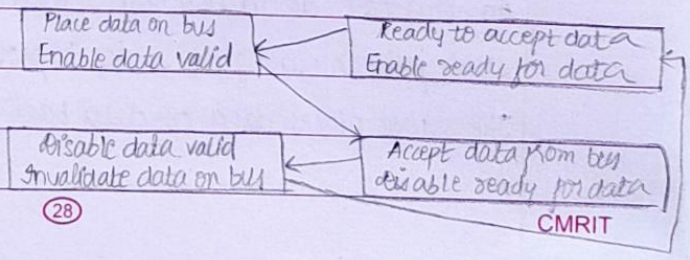


(ii) Destination initiated transfer

The destination unit disables its data accepted signal & system goes to initial state. The source does not send next data item until the destination shows its readiness to accept new data.







Asynchronous Communication Interface.

~~The block diagram is shown below~~. It functions as both transmitter & Receiver. The interface is initialized for a particular mode of transfer by means of a control byte that is loaded into its control register

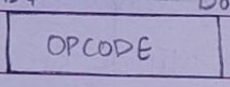
⑤ Explain about different instruction formats in 8086

INSTRUCTION FORMAT IN 8086 MICROPROCESSOR

Instruction format has one or more no. of fields.

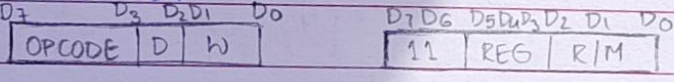
The first field is operation code (OR) OPCODE & the second field is operand field.

1) ONE BYTE INSTRUCTION



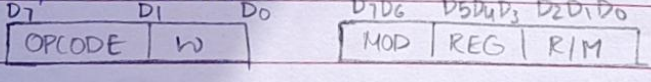
This format is only one byte long. The least significant 3 bits of opcode represent register operand otherwise all 8-bits are opcode bits & operands are implied.

2) REGISTER TO REGISTER



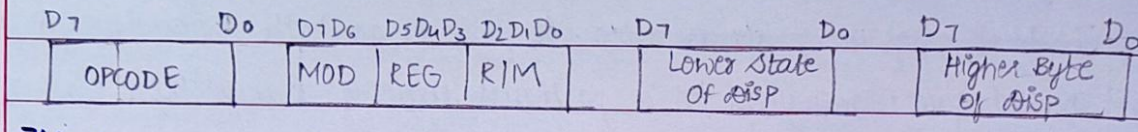
This format is 2 bytes long. The first byte of code specified the operation code and width of operand is specified by W. The second byte of code shows the register operands and R/M fields (specifies another register/location)

3) REGISTER TO/FROM MEMORY WITHOUT DISPLACEMENT



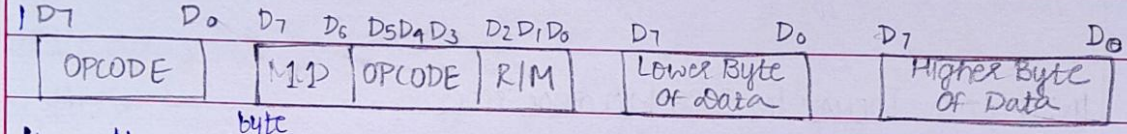
This format is also 2 bytes long & similar to register to register format except for MOD field that shows the mode of addressing.

4) REGISTER TO/FROM MEMORY WITH DISPLACEMENT



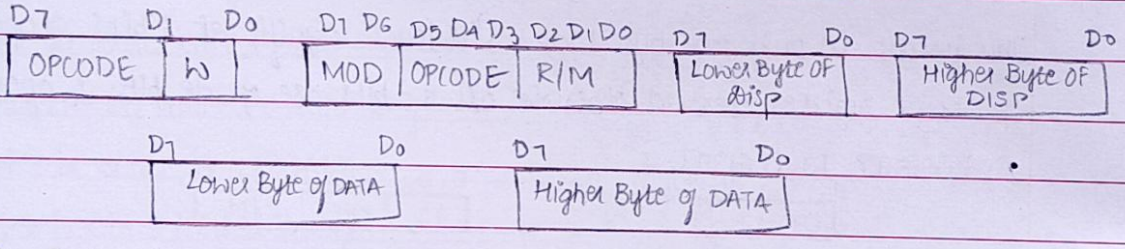
This type of format contains 1/two additional bytes for displacement along with 2 byte format of register to/~~register~~ from memory without displacement

5) IMMEDIATE OPERAND TO REGISTER



Here, the first byte as well as the 3-bits from second byte which are used for REG field in case of register to register format are used for opcode. It also contains one/two bytes of immediate data.

6) IMMEDIATE OPERAND TO MEMORY WITH 16-BIT DISPLACEMENT



This instruction format requires 5 or 6 bytes for coding. The first 2 bytes contain information regarding OPCODE, MOD and R/M fields. The remaining 4 bytes contain 2 bytes of displacement and 2 bytes of data.