Komar University of Science and Technology

College of Science

Subject: Computer Organization and Architecture– Spring 2020

Department Name: Computer Science

Title: Three Different Important Subjects Related to COA

Course Code: CPE2325

Course Name: Computer Organization and Architecture

Student ID: F180292

Student Name: Abdulrahman Tawffeq Jalal

Submission Date: 26-Jun-2020

Sequences	Assessment Categories	Description	Pages	40%
1	Title	Contains the required information shown in the cover page.		
2	Introduction	Briefly and properly introduces the topic.		
3	Body	Develops the topic properly.		
4	Conclusion	Is related to the introduction and the body and mentions the importance of the topic.		
5	References	3-5 sources related to the topic.		
	Organization	Is organized logically.		
	Length	2000 – 2500 words		

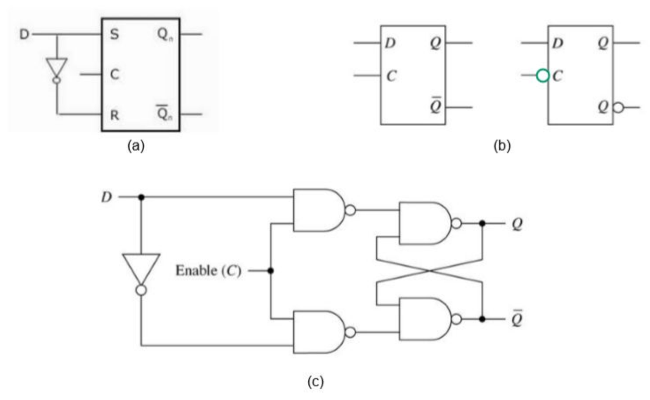
Contents

Introduction

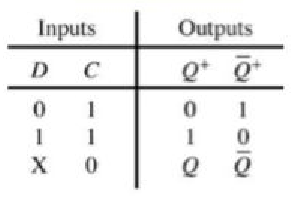
COA is the abbreviation of the computer organisation and architecture which is a very high science related to the computer. In this report, we will clarify three different kinds of necessary topics related to this science. The first part will be about a specific type of flip flops. How this flip flop works, transferring and storing data and its relationship with the time. The second part will be about the Bus and its important role of transferring data between registers. As well as, how it can be implemented. The third part will be about using assembly language which is a low level language (it is designed to deal with the computer more than it is designed to be programmed with). This language will be used to create a program to ask the user for some inputs then to produce a specific output.

Timing Diagram of Gated D Latch

The internal structure of gated D latch



The truth table of D flip-flop

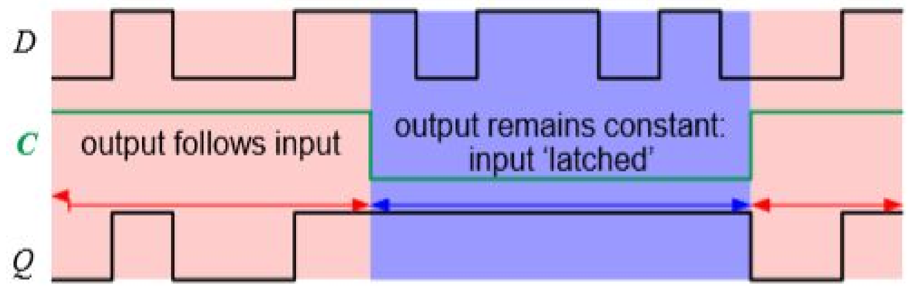


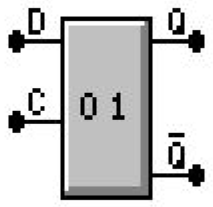
Inputs		Outputs	
D	C	Q^+	$\bar{Q}^+$
0	1	0	1
1	1	1	0
X	0	Q	$\bar{Q}$

NOTE: Because it is gated, that means it has a clock signal which can control the flip-flop. Once the clock is open, the output will be the same as the input.

The Timing Diagram

High & Low Level Triggered Type



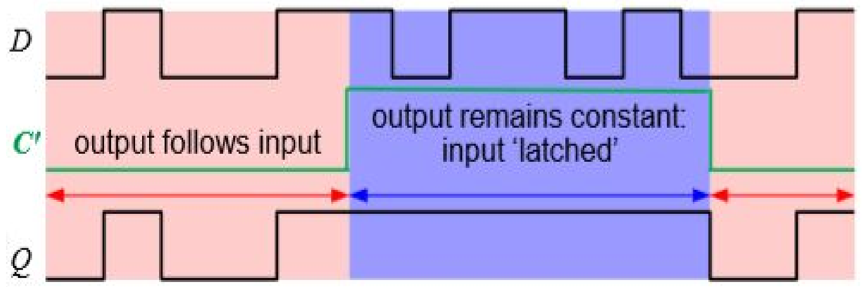


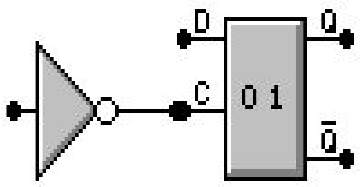
For this type, which is called a positive level triggered D flip flop, the output will follow the input and it will be equal to the input in case that the clock is equal to 1. It will work with the period and that's why it is called level triggered.

We can notice that in the first pink area the clock signal was 1, then the signal of the Q (output) is similar to the signal of the D (input).

In the beginning of the blue area, we can notice that the clock is equal to zero. Before that, the input was 1, then the output will be 1 and continue while the clock is off. The value 1 will be stored in the FF and we can notice the blue arrow in the diagram.

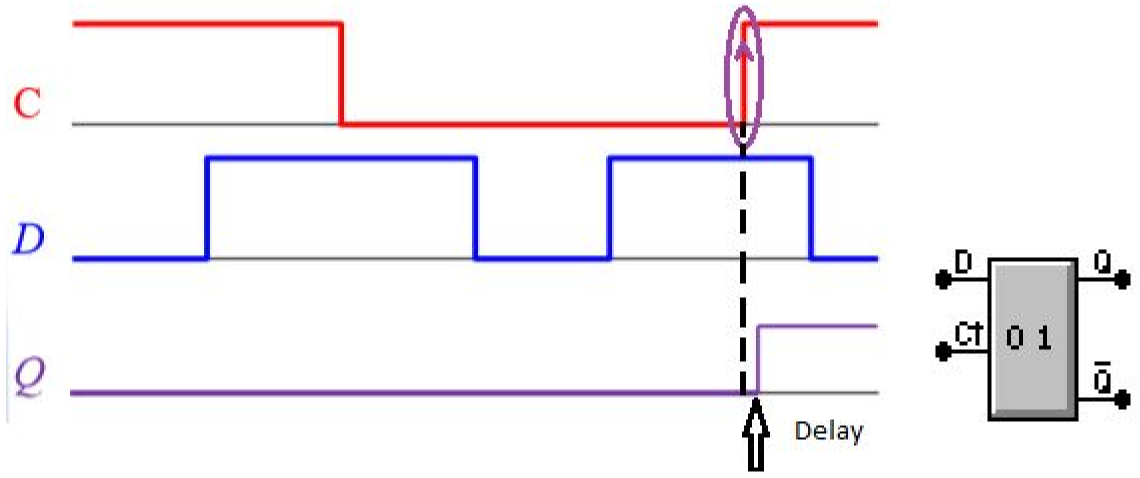
In the beginning of the second pink area, the clock became 1, so the output will be equal to the input during the period when the clock is equal to 1.





Here, we can notice that it is active low, the clock will work if it is equal to zero. So, we just inverted the clock signal to have the same result as the previous one.

Edge Triggered Type



We can notice that the output at the first is remained zero because this type of triggered FF is positive edge, meaning that the output will be affected and equal to the input only when the clock changes from zero to one (positive edge).

When there is a transition from zero to one in the signal of the clock, at that time the input D was 1, then the output changed to 1.

Note: The output will continue equal to one even if the input changes because it will be affected only during a very small period of time which is the positive edge. After that, it will continue with the same stored value till it faces the next positive edge to get the value of the input again during only this small position edge period.

Note: It is obvious that there was a delay in the output and this delay is called propagation delay.

Timing Delay

Propagation delay

The time required to change the output after applying the input and it is two types:

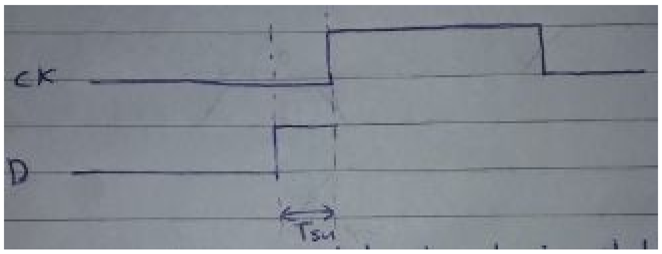
1. t PLH Low to High
2. t PHL High to Low

Minimum Pulse Width

The minimum amount of time a signal must be applied.

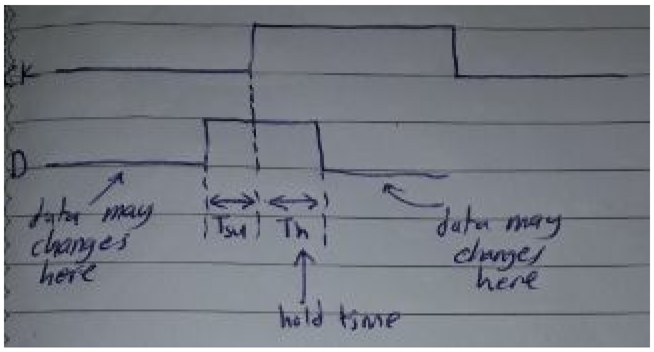
Setup Time

Is the minimum time interval for which the input signal must be stable.
How much time the data has to be stable, is defined by setup time. As well as, the data has to be stable during the hold time.



Hold Time

The minimum time interval for which the input signal must be stable following the sampling event of the clock for the input signal to be recognised correctly.



Note: To register the value properly into the flip flop, we have to make sure that the data is stable during the setup and hold time. So that the flip flop can sample the value and register it in the memory.

Note: If the data is not maintained to be stable during the setup and hold time, the flip flop then will get into a metastable state which is kind of undefined.

Bus Implementation

There are two ways to implement the Bus:

- Multiplexer
- Three-state Gates

We are going to explain the implementation of the Bus using Three-state Gates. But first we need to understand some terms.

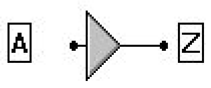
What is Bus?

Bus: It is the path for transferring information between registers. All the information that will be passed to the bus is either zero or one.

What is Buffer?

Buffer: We will explain about two types of it:

- The normal buffer: which in it the input will be equal to the out

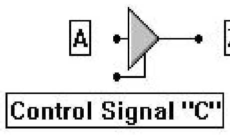


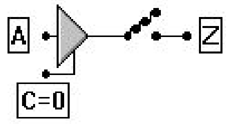
(The buffer equation: Z=A)

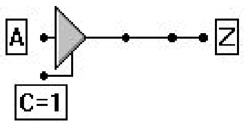
Truth Table:

A	Z
0	0
1	1

- The Tri-state Buffer, the output will not always equal to the input. There will be a control signal which will determine the result.







When C=0, it will be like an open circuit and this case is called High Impedance. In this case the Z will not have a value.

When the C=1, Z will be equal to A (Z=A) , means even zero or one.

It is called three-state gates because it will produces 0, 1, or High Impedance states

Truth table:

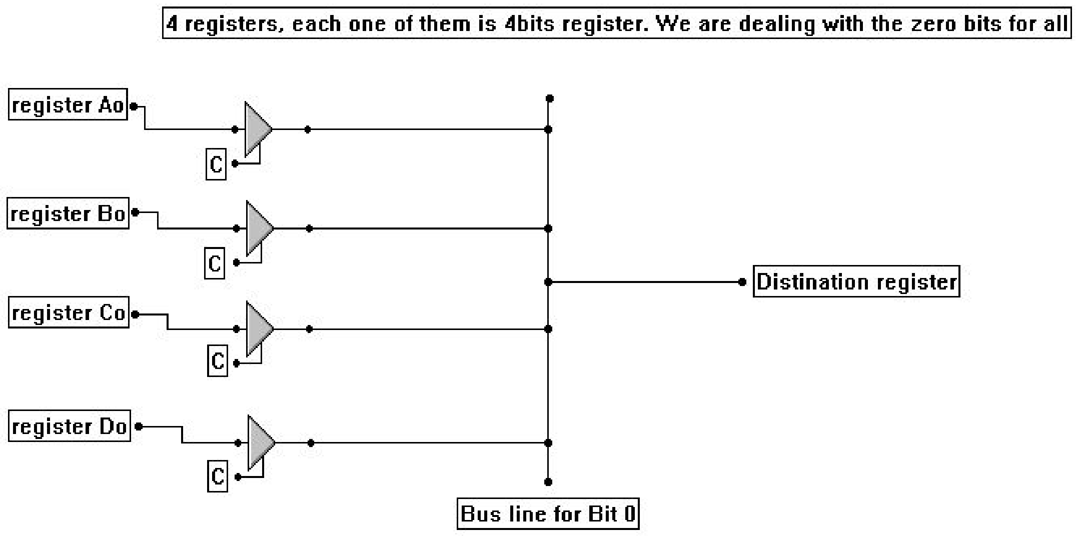
A(I/P)	C(Control)	Z(O/P)
1	1	1
0	1	0
1	0	High Impedance
0	0	High Impedance

High Impedance(HI)

It is a state when the output is not driven by the input, meaning that the output is not zero nor one. The output is disconnected from the circuit.

Control Signal

If the transferring is from register 1 to register 9 through the bus, the other registers should wait in order to transfer the data one by one (to avoid deformation of data). In this case, the control signal will do this task. it will deactivate the other registers when it is zero and will activate a specific register when it is equal to 1.


4 registers, each one of them is 4bits register. We are dealing with the zero bits for all
register Ao
C
register Bo
C
register Co
C
register Do
C
Distination register
Bus line for Bit 0

The bus line should have 1 bit at a time, in this case we need to activate just one of the control signals. To do such a thing, the decoder will be a good solution.

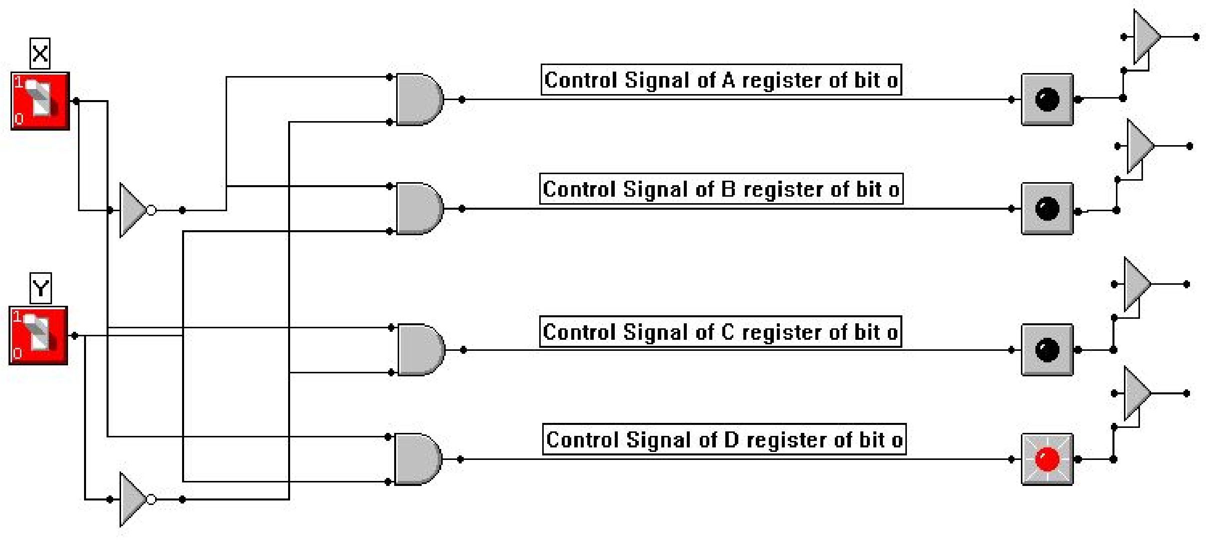
Decoder

It is a circuit which contains several AND gates connected in a way to activate one of the outputs and deactivate the others.

Example: 2 to 4 decoder which has 2 inputs(X, Y) and 4 outputs(will be the control signal of each register).
Truth Table:

X	Y	C(Ao)	C(Bo)	C(Co)	Z(Do)
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

So, we will activate one of the signals by changing the inputs of the decoder.


X
Y
Control Signal of A register of bit o
Control Signal of B register of bit o
Control Signal of C register of bit o
Control Signal of D register of bit o

As we can see by changing the inputs of the decoder to make them both ones, we could activate the control signal of Do register and deactivate the others. In this case zero or one from the Do register will go inside the bus line and the other will have a high impedance state.

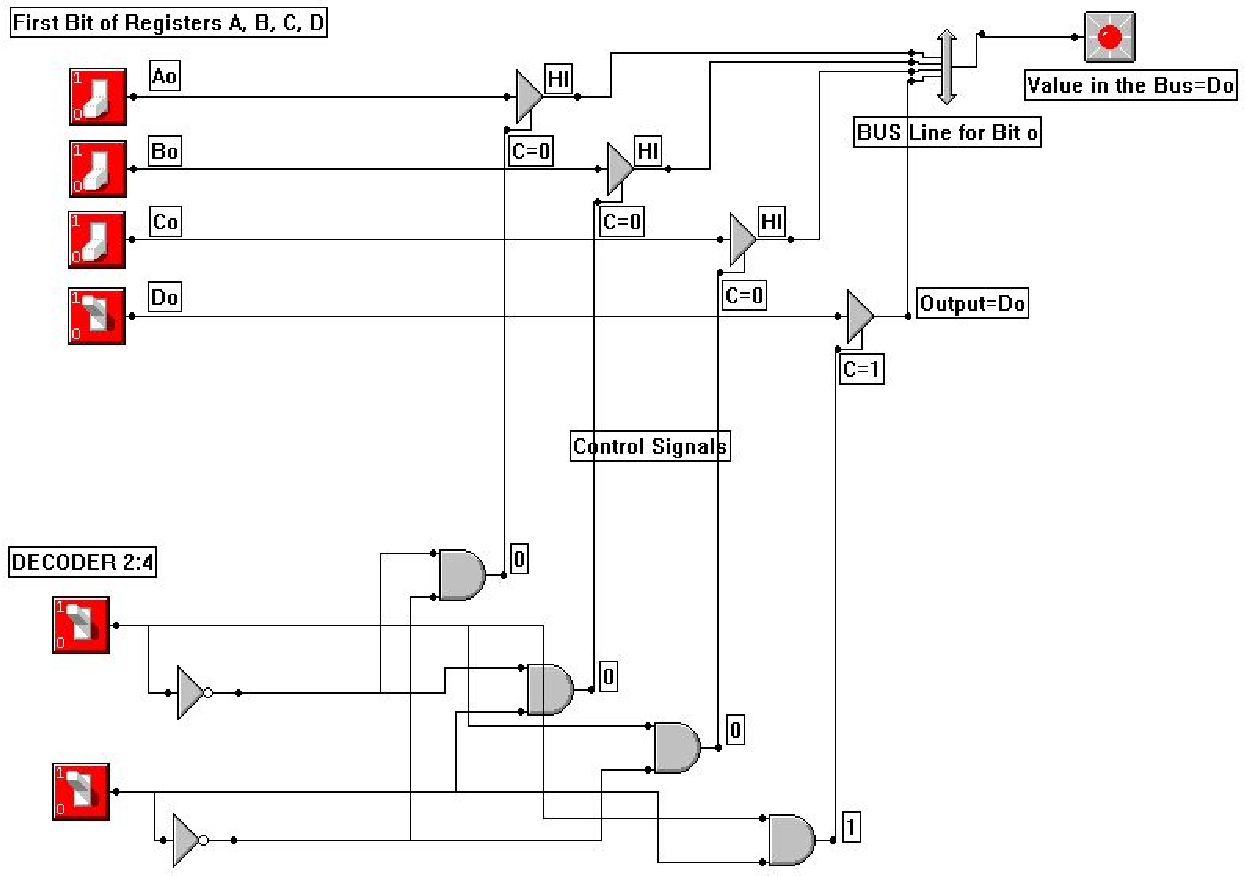
Implementation

Implementing the bus using the Three-state gates (for the 4-bit data transfer)

The requirements:

- Four registers of 4-bit type
- Tri-state buffer which has the control signal
- Bus Line which will have 1 bit at a time
- 2:4 decoder because we have 4 registers
- Switches
- LED
- Wires

NOTE: Because we will get either zero or one from each register, we will represent them with the switches.



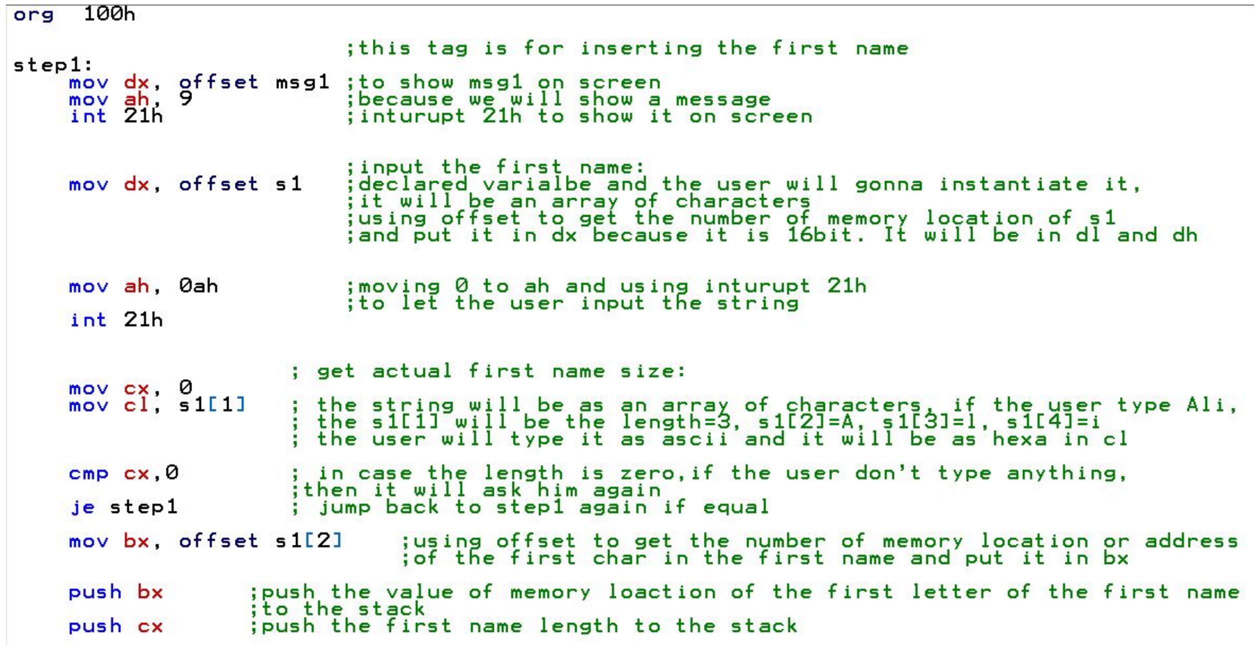
NOTE: The bus now has the value of the D register of its first bit and it is 1. If we switch it off, that means the value from the register will be zero and in this case the zero will be passed into the Bus.

NOTE: The 4 registers are 4-bit and here we only implemented it for the first bit which is (o).
That was for simplicity. If we want to have it as full to pass the 4 bits of each register, then we need to have 4 of this block above.

Assembly Language Project

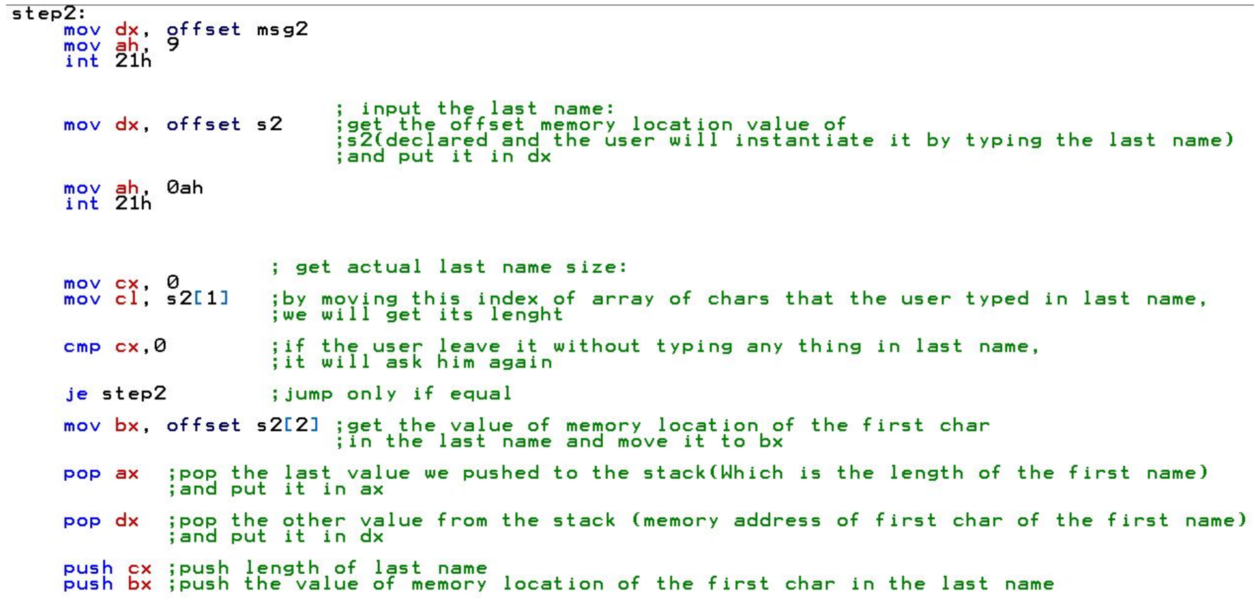
This project is a simple program using assembly language. It will ask the user to input his first and last names to show him his full name as a result.

This tag (Step1) will let the user be able to insert his first name:



```
org  100h
                              ;this tag is for inserting the first name
step1:
    mov dx, offset msg1 ;to show msg1 on screen
    mov ah, 9           ;because we will show a message
    int 21h             ;inturupt 21h to show it on screen


    mov dx, offset s1   ;input the first name:
                        ;declared varialbe and the user will gonna instantiate it,
                        ;it will be an array of characters
                        ;using offset to get the number of memory location of s1
                        ;and put it in dx because it is 16bit. It will be in dl and dh


    mov ah, 0ah         ;moving 0 to ah and using inturupt 21h
                        ;to let the user input the string
    int 21h


                       ; get actual first name size:
    mov cx, 0
    mov cl, s1[1]      ; the string will be as an array of characters, if the user type Ali,
                       ; the s1[1] will be the length=3, s1[2]=A, s1[3]=l, s1[4]=i
                       ; the user will type it as ascii and it will be as hexa in cl

    cmp cx,0           ; in case the length is zero,if the user don't type anything,
                       ;then it will ask him again
    je step1           ; jump back to step1 again if equal

    mov bx, offset s1[2]    ;using offset to get the number of memory location or address
                            ;of the first char in the first name and put it in bx

    push bx      ;push the value of memory loaction of the first letter of the first name
                 ;to the stack
    push cx      ;push the first name length to the stack
```

This tag (step2) will let him insert the last name:



```
step2:
    mov dx, offset msg2
    mov ah, 9
    int 21h


                        ; input the last name:
    mov dx, offset s2   ;get the offset memory location value of
                        ;s2(declared and the user will instantiate it by typing the last name)
                        ;and put it in dx

    mov ah, 0ah
    int 21h


                   ; get actual last name size:
    mov cx, 0
    mov cl, s2[1]  ;by moving this index of array of chars that the user typed in last name,
                   ;we will get its lenght

    cmp cx,0       ;if the user leave it without typing any thing in last name,
                   ;it will ask him again

    je step2       ;jump only if equal

    mov bx, offset s2[2] ;get the value of memory location of the first char
                         ;in the last name and move it to bx

    pop ax   ;pop the last value we pushed to the stack(Which is the length of the first name)
             ;and put it in ax

    pop dx   ;pop the other value from the stack (memory address of first char of the first name)
             ;and put it in dx

    push cx ;push length of last name
    push bx ;push the value of memory location of the first char in the last name
```

This is also in the same tag (step2) which will call for the other tags to print the full name:

```
mov cx,ax ;mov the length of first name to cx
mov bx,dx ;mov the value of memory loaction address of the first char in first name to bx
          ;we did this move because we are going to show a mesg,
          ;so we dont want the values to be changed as we will use the dx and ax

mov dx, offset msg4
mov ah, 9
int 21h

call loop1            ;to print the first name char by char


mov dx, offset space  ;to print a space between first and last names
mov ah, 9
int 21h


pop bx   ;pop the last value from the stack (memory address of first char of the last name)
         ;and put it in bx
pop cx   ;pop the other value we pushed to the stack(Which is the length of the last name)
         ;and put it in cx

call loop2   ;to print the last name char by char
```

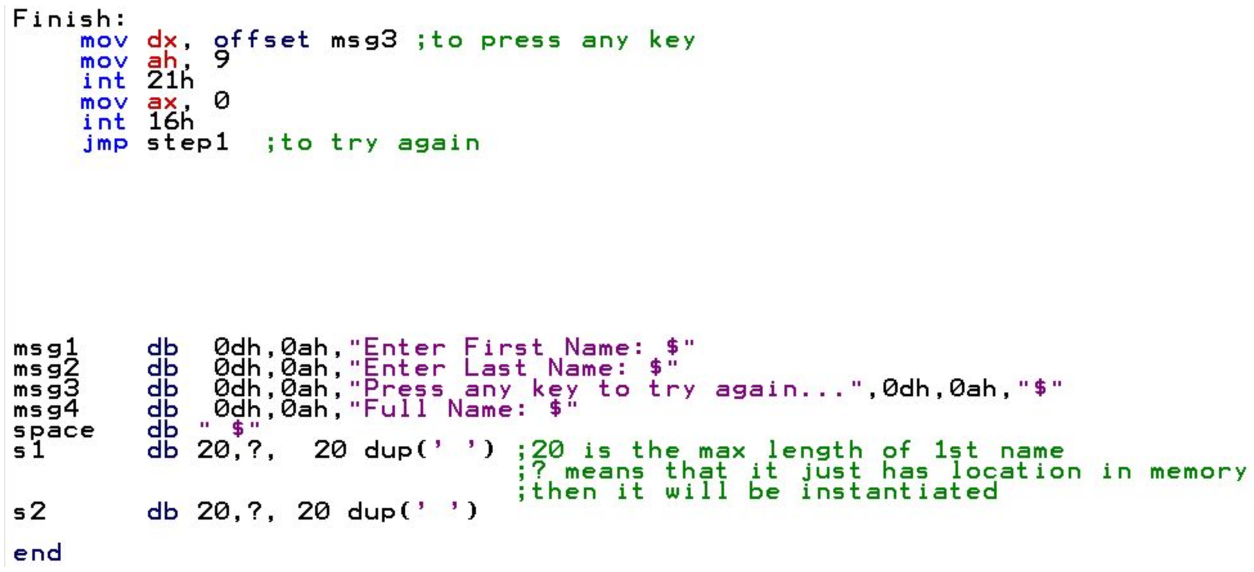
This tag is for printing the first name of the user:

```
                   ;to print the first name characters
loop1:
    print_char:
    mov dl, [bx] ;by put the value of memory location of the first char in first name into [ ],
                 ;it will point to the first char itself
    mov ah, 2    ;to display the char on the screen, we should mov 2.
                 ;if it is msg we should mov 9 to ah
    int 21h

    cmp al,20h   ;compare if it is space, to return back to print the last name.
                 ;We only want the first name and any space means that
                 ;the user typed the middle name as well

    je return    ;if there is a space, means he finished the first name,
                 ;so we will return back to the place after call loop1
                 ;to have a space then to print the last name

    inc bx           ;it will increment the value of the memory loaction address of char1
                     ;to get the place of char 2

    loop print_char  ;loop to print the second char of first name
                     ;the number of times of loop will be according to the length of the
                     ;first name in cx
                     ;if cx=0, means all chars are printed out and it will return back to
                     ;the place after call loop1



return:
    ret
```

This tag is for printing the last name:

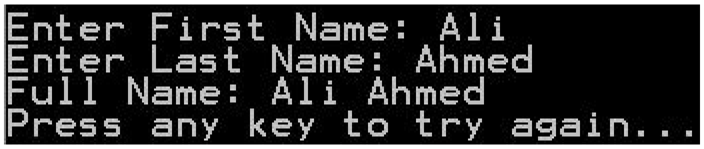
```
                          ;to print the last name characters
loop2:
    print_char2:
    mov dl, [bx]
    mov ah, 2
    int 21h
    cmp al,20h
    je return
    inc bx
    loop print_char2      ;the number of times of loop will be according to the length of
                          ;the last name in cx
                          ;if cx=0, means all chars are printed out

    jmp Finish            ;to finish tag
```

The end of the project:



```
Finish:
    mov dx, offset msg3 ;to press any key
    mov ah, 9
    int 21h
    mov ax, 0
    int 16h
    jmp step1  ;to try again





msg1     db  0dh,0ah,"Enter First Name: $"
msg2     db  0dh,0ah,"Enter Last Name: $"
msg3     db  0dh,0ah,"Press any key to try again...",0dh,0ah,"$"
msg4     db  0dh,0ah,"Full Name: $"
space    db " $"
s1       db 20,?,  20 dup(' ') ;20 is the max length of 1st name
                               ;? means that it just has location in memory
                               ;then it will be instantiated
s2       db 20,?, 20 dup(' ')

end
```

The result:



```
Enter First Name: Ali
Enter Last Name: Ahmed
Full Name: Ali Ahmed
Press any key to try again...
```

Conclusion

In conclusion, we note from what has been presented in this report that computer organization and architecture is a science related to the hardware part of a computer. It also shows us the extent and magnitude of this field. What we noticed in this report on how to transfer and store data between the hardware parts of a computer, input and output and its relationship to time, and how the machine language deals with the computer is only a small part of this broad knowledge.

References

1. https://www.youtube.com/channel/UCyMvGzplDSfzaqyqs6oKT3A
2. https://www.youtube.com/user/rasmurtech
3. https://www.robotbrigate.com
4. https://www.tutorialspoint.com/videotutorials/index.php
5. https://www.youtube.com/channel/UCfwtsVNshp2PX4M9tNSnDpw