A picture containing text, clipart

Description automatically generated

**——————————————————————————————————**

**SECR 2033**

**COMPUTER ORGANISATION AND ARCHITECTURE**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**GROUP PROJECT**

SECTION : 01

LECTURER : DR. ZURIAHATI BINTI MOHD YUNOS

PREPARED BY : GROUP 03

GROUP MEMBERS :

|  |  |  |
| --- | --- | --- |
| NO. | NAME | MATRIC NO. |
| 1. | MEK ZHI QING | A20EC0077 |
| 2. | CHANG MIN XUAN | A20EC0024 |
| 3. | KESHINIY A/P MOGAN | A20EC0057 |
| 4. | ZEREEN TEO HUEY HUEY | A20EC0173 |

VIDEO LINK: <https://youtu.be/HMcGeHG-O9Q>

**Table of Content**

Member Responsibilities………………………………………………………………1

Coding and Explanation……………………………………………………………….2

Example of Inputs and Outputs……………………………………………………......9

Discussion and Conclusion……………………………………………….………….11

References……………………………………………………………………………12

Appendix……………………………………………………………………………..12

**Member Responsibilities**

|  |  |
| --- | --- |
| **Name** | **Description** |
| Mek Zhi Qing | Program:   * Binary to decimal * Bug solving * Check and link all the function together   Report:   * Example of inputs and outputs * Coding and explanation |
| Chang Min Xuan | Program:   * Looping function of the menu   Report:   * Coding and explanation |
| Keshiniy A/P Mogan | Program:   * Decimal to binary   Report:   * Discussion and conclusion |
| Zereen Teo Huey Huey | Program:   * Binary to decimal   Report:   * Coding and explanation |

**Coding and Explanation**

In this project, we are required to apply our knowledge of assembly language that we had learned in the subject computer organisation and architecture. We use Visual Studio as a platform to conduct our project. Based on the task given by the lecturer, we are required to write an assembly program which can do the conversion function from 8-bit unsigned binary to decimal and from positive decimal to 8-bit binaries.

Scatter chart

Description automatically generated with medium confidence

Figure 1: Define the variable

First, we create the variables based on their data types. The question to be displayed is store in the variable of str 1 until str 14 which will be call later when the program is run to display it to the user. The others variable is uninitialized for the further usage in the program.

Table

Description automatically generated with low confidence

Figure 2: L1 loop

In the main program, L1 acts as a loop to display the menu of the conversion program. It will display the question to ask the user whether they want to do the conversion of 8-bit unsigned binary to decimal, conversion of positive decimal to 8-bit binaries or end the program. If the user enters 1, it will jump to the loop named BtoD to carry out the conversion of 8-bit unsigned binary to decimal. However, if the user enters 2, it will jump to the loop named DtoB to carry out the conversion of positive decimal to 8-bit binaries; if user enter 3, it will jump to the loop named Bye which leads to the end of the program. This process is done by using the function named cmp where the data enter by the user will be compared with the constant we set in the program.

Text

Description automatically generated

Figure 3: BtoD loop

Text

Description automatically generated

Figure 4: L2 loop

In the BtoD loop, the user is required to enter the 8-bit binary digits. After the statement of the conversion is display, it will jump to L2 for the further conversion process. In L2, the value enter by the user is now store in eax, the cmp function is used to compare it with 0. If eax is equal to 0, it will jump to L3; if not, the program will continue to run the code in L2. The value will divide by 10 by using the function div. Next, the remainder is push into stack. After that, the xor is used to restore the edx with 0. The value of ecx is increase 1 for the further use as a counter. This process is continue looping until the eax is equal to 0 and jump to L3.

Text

Description automatically generated

Figure 5: L3 loop

In L3 loop, the ecx is compared to 0 and if it is equal to 0 then the program will jump to L8. In contrast, it will continue in L3, and the remainder is pop from the stack and then compare with 0. If the value pop out is equal to 0, it will jump to L5 to decrease the ecx and counter while if the edx is equal to 1, it will continue the process in L3 which is move the value of edx to eax and move the ecx to count as the counter to be use in L6 loop. Then, the process will continue in L6.

A picture containing graphical user interface

Description automatically generated

Figure 6: L5 loop

In L5 loop, this loop is used when the remainder pop out from edx is 0. The ecx and counter will decrease by 1 and it will then return to L3 loop for the further conversion.

Text

Description automatically generated

Figure 7: L6 loop

In L6 loop, the value in the count variable will compare to 1 and if it is equal to 1 the program will jump to L7 loop. If not equal, the program will multiply the value with 2. Then, the count will decrease by 1. The looping will be continuing to carry out all these steps until the count is equal to 1 to gain the decimal number. Then, the program will continue the process in L7 loop.

Text

Description automatically generated

Figure 8: L7 loop

In L7 loop, the value we obtain in decimal from L6 loop will be moved into total. The value of total will move to eax and add the eax value to dValue. Next, the counter and ecx value will decrease 1. After that, the process will continue by jumping back to L3 loop to pop the second number in stack from edx. All the steps will keep repeated until all 8-bit binary is done.

Text, letter

Description automatically generated

Figure 9: L8 loop

When the 8-bit binary conversion is done, the process will go to L8 loop. The value in dValue will move to eax and display out by the function call writeDec. After the value is display, the process will jump to L1 loop to display the menu again.

A picture containing text

Description automatically generated

Figure 10: DtoB loop

In DtoB loop, the user is prompted to enter a decimal integer value by using the function readDec. Then, writeDec is called to display the decimal integer value entered at the end of string 11. It will then jump to the next loop.

Text, letter

Description automatically generated

Figure 11: next loop

In next loop, the decimal integer value previously entered by the user is compared with 0 by using cmp function. If eax value is equals to 0, it will jump to zero loop, or else, it will continue the loop. Then, the value is divided by 2, and the remainder is pushed into stack and stored in edx. The value of ecx is incremented by one. XOR instruction is used to set edx to 0. The looping will continue unless eax is equal to 0, then it will jump to zero loop.

Text, letter

Description automatically generated

Figure 12: then loop

In then loop, the value of ecx is compared with 0. If ecx equals to 0, it will jump to last loop. If not equal, the remainder from stack will be pop out and move to eax. The function call writeDec is used to display the value stored in eax. Then, ecx is decremented by one. The looping will continue until ecx is equal to 0, then it will jump to last loop.

A picture containing table

Description automatically generated

Figure 13: zero loop

In zero loop, ecx is compared with 8. If ecx equals to 8, it will jump to then loop. If not equal, 0 is moved into variable named value and it will be pushed into the stack and stored in edx. The ecx value is increased by one. The looping will continue until ecx equals to 8 and jump to then loop.

Text

Description automatically generated

Figure 14: last loop

In last loop, string 13 is called and displayed at the output. Then, the looping ends and jump to L1 loop again to display menu.

Graphical user interface, text

Description automatically generated

Figure 15: Bye loop

In Bye loop, string 12 will be called and displayed at the output. Then the program ends.

**Example of Inputs and Outputs**

A picture containing text, screenshot, monitor, computer

Description automatically generated

*Example of inputs and outputs when the user choose 1 (convert binary to decimal)*

Graphical user interface, text

Description automatically generated

*Example of inputs and outputs when the user choose 2 (convert decimal to binary)*

A screenshot of a computer

Description automatically generated with medium confidence

*Example of inputs and outputs when the user choose 3*

**Discussion and Conclusion**

Since it is a pandemic time and we have to stay at our own place, we carried out this project fully online. Even though we stayed apart, we were still determined to complete this project successfully with a great teamwork. We discussed about project through WhatsApp and decided our responsibilities on this project. We separated our responsibilities equally for both program and report. Since there was not a face-to-face communication, we faced difficulties when doing the program. For example, we have to wait for the looping and menu coding to finish in order to complete the coding part for Decimal to Binary and Binary to Decimal. Since we planned to complete this project before our final exams, it was quite stressful when we have to complete other course projects too. Fortunately, we were able to complete our project which includes the full coding, report, and a video before the deadline with the spirit of teamwork.

In order to complete this project, we applied our understandings and knowledge that we learnt from Computer Organization and Architecture which was taught by Dr Zuriahati Bt. Mohd Yunos. For example, we used many instructions such as CALL, JMP and MOV in order to execute the coding. We also used BYTE to declare all the strings and DWORD to declare the variables which will hold a value. Not only that, but we also used loop in order to repeat the process in the loop. Using Reverse Polish Nation, RPN helped us to change decimal value into binary and binary into decimal by pushing in and popping out the remainder from the stack. At both conversions part, the XOR is used to restore the edx to 0. Last but not least, the converted value is displayed with proper sentence.

Last but not least, this project helped us to increase our understandings and knowledge in assembly language programming. Besides, it also gave us a chance to know and understand our groupmates more. Since programming languages are very important in developing software system or programs, it is important to understand assembly language. Lastly, we would like to thank Dr Zuriahati for teaching and helping us in the Computer Organization and Architecture course.

**References**

Pearson Education, 2010, Assembly Language for x86 Processors 6th Edition (Chapter 6), Kip R Irvine, last accessed on 2 July 2021

Stack Overflow, Convert binary to decimal,

<https://stackoverflow.com/questions/33426296/how-can-i-convert-binary-to-decimal-in-8086-assembly-nasm>, last accessed on 2 July 2021

**Appendix**

TITLE Project(main.asm)

; created on 28/6/2021

INCLUDE Irvine32.inc

.data

str1 BYTE ">>>Please select the conversion type:", 0

str2 BYTE "1. Binary to Decimal", 0

str3 BYTE "2. Decimal to Binary", 0

str4 BYTE "3. Exit", 0

str5 BYTE "--------------------------------------", 0

str6 BYTE "Enter your choice: ", 0

str7 BYTE "Please Enter 8-bit binary digits (e.g., 11110000): ", 0

str8 BYTE "The decimal integer of ", 0

str9 BYTE " is ", 0

str10 BYTE "Please Enter a decimal integer less than 256: ", 0

str11 BYTE "The binary of ", 0

str12 BYTE "Bye.", 0

str13 BYTE "b",0

str14 BYTE "d",0

dValue DWORD ?

total DWORD 0

counter DWORD ?

count DWORD ?

totald DWORD ?

value DWORD ?

.code

main PROC

; display menu

L1 :

mov edx, offset str1

call writestring

call Crlf

mov edx, offset str2

call writestring

call Crlf

mov edx, offset str3

call writestring

call Crlf

mov edx, offset str4

call writestring

call Crlf

mov edx, offset str5

call writestring

call Crlf

mov edx, offset str6

call writestring

call ReadDec

cmp eax, 1

je BtoD

cmp eax, 2

je DtoB

cmp eax, 3

je Bye

BtoD :

mov edx, offset str7

call writestring

call readDec

mov edx, offset str8

call writestring

call writeDec

mov edx, offset str13

call writestring

mov edx, offset str9

call writestring

mov ecx,0

mov edx,0

jmp L2

L2:

cmp eax,0

je L3

mov ebx,10

div ebx ;divide the value by 10

push edx ;push the remainder into stack

xor edx,edx

inc ecx

mov counter,ecx

mov dValue,0

jmp L2

L3:

cmp ecx,0

je L8

pop edx ; pop the remainder from stack

cmp edx,0 ; compare the remainder with 0

je L5

mov eax,edx

mov count, ecx

jmp L6

;when the value pop out is equal to zero

L5:

dec ecx

dec counter

jmp L3

;when the value pop out is equal to one

L6:

cmp count,1

je L7

mov ebx,2

mul ebx

dec count

jmp L6

L7:

mov total,eax

mov eax,total

add dValue,eax

dec counter

dec ecx

jmp L3

L8:

mov eax, dValue

call writeDec

mov edx, offset str14

call writestring

call Crlf

call Crlf

jmp L1

;Decimal to Binary

DtoB:

mov edx, offset str10

call writestring

call readDec

mov edx, offset str11

call writestring

call writeDec

mov edx, offset str14

call writestring

mov edx, offset str9

call writestring

mov ecx,0

mov edx,0

jmp next

next:

cmp eax,0

je zero

mov ebx,2

div ebx ;div the value by 2

push edx ;push remainder into stack

inc ecx

xor edx,edx ;mov 0 to dx

jmp next

then:

cmp ecx,0

je last

pop edx ; pop remainder from stack

mov eax,edx

call writeDec

dec ecx

jmp then

zero:

cmp ecx, 8

je then

mov value,0

push edx

inc ecx

jmp zero

last:

mov edx, offset str13

call writestring

call Crlf

call Crlf

jmp L1

Bye:

mov edx, offset str12

call writestring

exit

main ENDP

END main