# SECR 2033 Computer Organization and Architecture 2019/2020-2

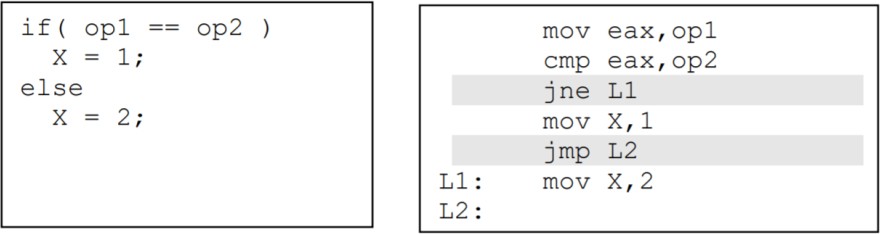
**Programming 6: CONDITIONAL STRUCTURES**

**HAM JING YI A19EC0048 SECTION 2**

## Part A – Programming review

1. **Block-Structured IF Statements**

Assembly language programmers can easily translate logical statements written in C++/Java into assembly language



# Compound Expressions

Logical AND Operator

When implementing the logical AND and OR operator, consider that high-level languages compilers for Java, C, and C++ use short-circuit evaluation for efficiency reasons.

*The second expression is not evaluated if the first expression is false. (early exit)*

|  |  |
| --- | --- |
|  |  |

Logical OR Operator

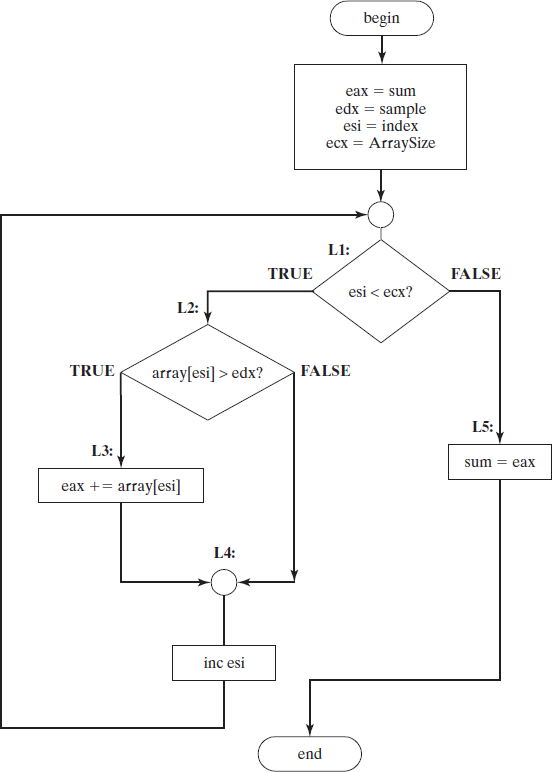
|  |  |
| --- | --- |
|  |  |

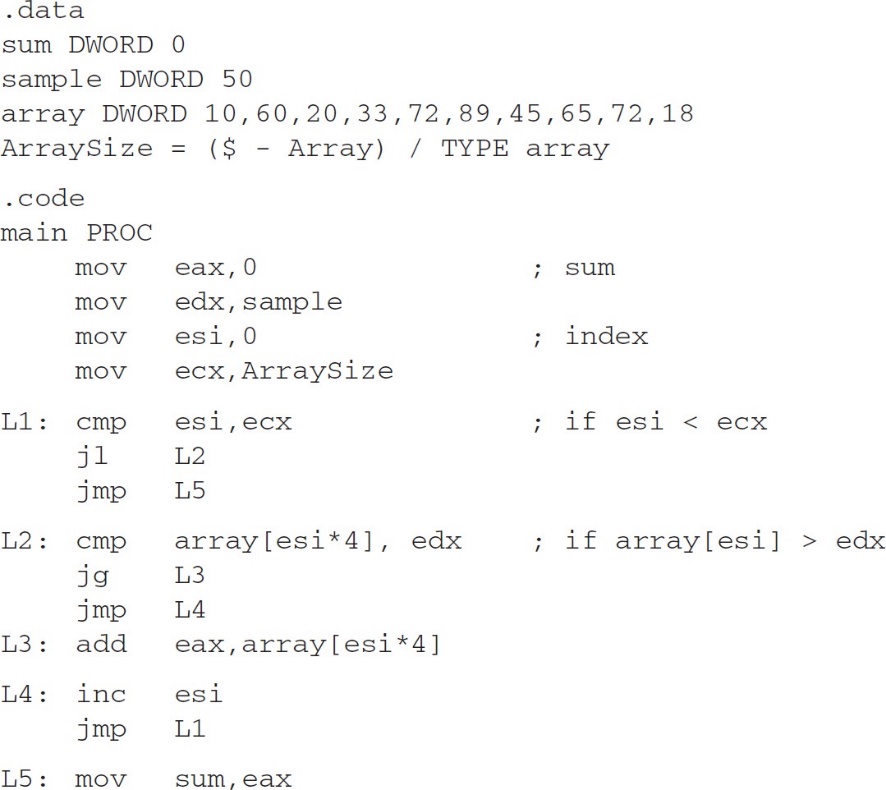
# WHILE Loops

The WHILE structure tests a condition first before performing a block of statement. As long as the loop condition remains true, the statement is repeated.

|  |  |
| --- | --- |
|  |  |

# Assembly Code

The easiest way to generate the assembly code from a flowchart is to implement the code for each shape. Note the direct correlation between the flowchart labels and the labels used in the source code.



## Part B – Let’s do a little programming on your own

* 1. Implement the following pseudocode in assembly language, using unsigned values. Please use short-circuit evaluation in your code.

IF (MY\_MONEY < YOUR\_MONEY) YOU\_DONATE = 20;

ELSE

{

YOU\_DONATE = 10;

I\_DONATE = 10;

}

ANSWER:

|  |
| --- |
| mov eax, MY\_MONEY  cmp eax, YOUR\_MONEY  jb L1  mov YOU\_DONATE, 10  mov I\_DONATE, 10  jmp L2  L1: mov YOU\_DONATE,20  L2: |

* 1. Implement the following pseudocode in assembly language, using unsigned values. Please use short-circuit evaluation in your code.

IF (EBX > ECX && ECX <= EDX)

{

EAX = 9;

EDX = 10;

}

ANSWER:

|  |
| --- |
| cmp ebx, ecx  jbe next  cmp ecx, edx  ja next  mov eax, 9  mov edx, 10  next : |

* 1. Implement the following loop, using unsigned 32-bit integers. Please use the reverse condition to make the code shorter.

WHILE (MINE <= MIN\_CONTRIBUTION)

{

VAR++;

PRINTF(“Sorry your contribution is not enough.”);

}

PRINTF(“Thank you for your generous contribution.”);

ANSWER:

|  |
| --- |
| .data  MINE DWORD ?  MIN\_CONTRIBUTION DWORD ?  VAR DWORD ?  str1 BYTE "Sorry your contribution is not enough.",0  str2 BYTE "Thank you for your generous contribution.",0  .code  main PROC    top :  mov eax, MINE  cmp eax, MIN\_CONTRIBUTION  ja next  inc VAR  mov edx,OFFSET str1  call WriteString  call crlf  jmp top  next :  mov edx,OFFSET str2  call WriteString  call crlf |

* 1. Rewrite the code from section (D) so it is functionally equivalent but uses fewer instructions.

ANSWER:

|  |
| --- |
| .data  sum DWORD 0  sample DWORD 50  array DWORD 10,60,20,33,72,89,45,65,72,18  ArraySize = ( $ - Array)/ TYPE array  .code  main PROC  mov eax, 0  mov edx, sample  mov esi, 0  mov ecx, ArraySize  L1: cmp esi,ecx  jge L5  cmp array[esi \* 4], edx  jle L4  add eax, array[esi \* 4]  L4: inc esi  jmp L1  L5: mov sum, eax |

* 1. Suppose a student wants to decide to go home or stay in UTM during online teaching & learning phase. Two criteria are used to determine whether the student will go home or not:
     + First is the student’s hometown current zone, whether it is in green or red zone.
     + Second is the internet speed in the student’s hometown. The speed must at least be 2MBps to have a decent teaching & learning experience.

1. Draw a flowchart of the program.

End

Print “go home”

ebx=speed

ecx=zone

ANSWER:

Begin

Green zone =1, red zone=2, other =3

Print “stay in UTM”

ecx==1 && ebx>=2

L1 :

FALSE

TRUE

1. Write an assembly language program that ask the user to enter the inputs and suggest a decision to the student.

ANSWER:

|  |
| --- |
| TITLE MASM Template (main.asm)  ; Description:  ;  ; Revision date:  INCLUDE Irvine32.inc  .data  str1 BYTE "Please enter your hometown current zone (green=1, red=2, other=3): ",0  str2 BYTE "Please enter your internet speed in your hometown (MBps) : ",0  str3 BYTE "Decision : GO HOME",0  str4 BYTE "Decision : STAY IN UTM",0  .code  main PROC  mov edx,OFFSET str1  call WriteString  call ReadDec  mov ecx,eax  mov edx,OFFSET str2  call WriteString  call ReadDec  mov ebx, eax  cmp ecx,1  jne L1  cmp ebx,2  jb L1    mov edx,OFFSET str3  call WriteString  call crlf  jmp outt  L1 :  mov edx,OFFSET str4  call WriteString  call crlf  outt:  exit  main ENDP  END main |