National University of Computer and Emerging Sciences
Karachi Campus

Computer Organization and Assembly language

Date: September 24th 2024

Course Instructor(s)

Mr. Shoaib Rauf, Mr.Kashan Hussain, Mr. Aashir Mahboob, Ms. Atiya Jokhio, Mr. M.Kariz Kamal, Mr. M.Usman, Mr. Nauraiz Subhan

Sessional-I Exam

Total Time (Hrs): 1
Total Marks: 30
Total Questions: 3

Roll No Section Student Signature

Do not write below this line

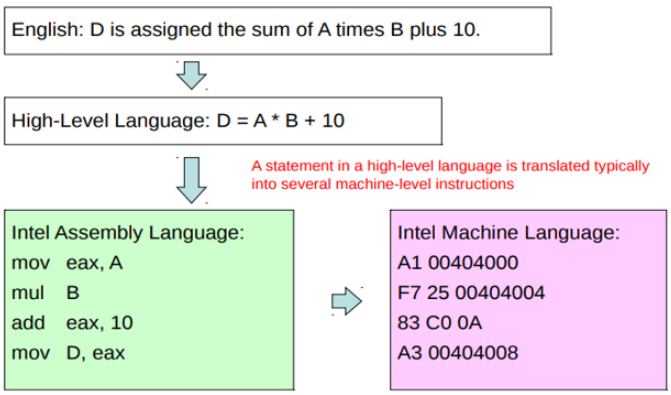
Attempt all the questions.

***CLO #1 & 2:* Illustrate micro-architecture of x86 and RISC processors. Create basic assembly code using different type of addressing modes in x86 and RISC ISA(s) to solve simple-moderate problems.**

Q1: Answer the following questions. **[Marks: 7*2 = 14]**

I. What is meant by a one-to-many and one-to-one relationship in context of x86 assembly. Use a coding example to support your answer.

A single statement in C++ expands into multiple assembly language or machine instructions. (***one-to-many* relationship**)
Each assembly language instruction corresponds to a single machine-language instruction. (***one-to-one* relationship)**



II. Explain the Direct, In-Direct and Indexed Modes of accessing a memory in x86 assemblies using an example instruction.

. A direct memory operand is a named reference to storage in memory:

```
.data
var1 BYTE 10h
.code
```

```
mov al,var1
```

An indirect operand holds the address of a variable, usually an array or string. It can be dereferenced (just like a pointer).

```
.data
val1 BYTE 10h,20h,30h
.code
mov esi,OFFSET val1
mov al,[esi]
```

An indexed operand adds a constant to a register to generate an effective address.

```
.data
arrayW WORD 1000h,2000h,3000h
.code
mov esi,0
mov ax,[arrayW + esi]
```

III. Briefly explain the role of Virtual Machine(s) in execution of a high level language code.

The high level code is compiled to Java byte code and then the JVM executes the code on the hardware platform that is running.

IV. Why Assembly language is not portable?

Assembly language is not portable because it is designed for a specific processor family.

V. Explain the difference(s) between Real and Protected modes of x86 processors.

Real Mode

only 1 MB of memory can be accessed from 0 to FFFFF
Program can access any part of main memory
MS-DOS runs in real-address mode

Protected Mode

Each program can access a maximum of 4GB of memory
OS assign memory to each running program
Program are prevalent from accessing each other memory
Windows NT, 2000, XP and Linux used Protected mode

VI. What is the role of memory segmentation in x86 Intel processors regarding mismatch in register size and address bus size?

Segmentation is the process in which the main memory of the computer is logically divided into different segments and each segment has its own base address. It is basically used to enhance the speed of execution of the computer system. It allows to extend the address ability of the processor, i.e. segmentation allows the use of 16 bit registers to give an addressing capability of 1 Megabytes. Without segmentation, it would require 20 bit registers.

VII. How many clock cycles are required in order to read a single value from main memory upon CPU request?

This task requires four steps and each step requires at least one clock cycle to execute.

CLO # 2: **Create basic assembly code using different type of addressing modes in x86 and RISC ISAs to solve simple-moderate problems.**

Q2: Using the following **variable definitions answer the questions given below.** **[Marks: 5+5 = 10]**

```
.data
    var1 SBYTE -4,-2,3,1
    var2 WORD 1000h,2000h,3000h,4000h
    var3 SWORD -16,-42
    var4 DWORD 1,2,3,4,5
```

I. Draw a memory map and assign proper physical addresses (using a real address mode) to each byte stored in the above data segment (Assume DS= 0040h, and the starting offset is 0DEFh).

Physical Address= 00400 + 0DEF = 011EFh

011EFh	-4	011FBh	-16
011F0h	-2	011FDh	-42
011F1h	3	011FFh	1
011F2h	1	1203h	2
011F3h	1000h	1207h	3
011F5h	2000h	120Bh	4
011F7h	3000h	120Fh	5
011F9h	4000h		

II. What will be the hexadecimal value(s) of the destination operand after each of the following instructions are executed in sequence?

a. **mov ah, varl** **FCh**
b. **mov ah, byte ptr[var2+3]** **20h**
c. **movzx ebx, var2** **00001000h**
d. **xchg cx, [var2+4]** **3000h**
e. **mov ax, var3** **FFF0h**
f. **mov ax,[var3-2]** **4000h**
g. **movsx, ebx, word ptr var4** **00000001h**
h. **add cx, var3[type var4]** **3001h**
i. **mov al, [varl+6]** **0h**
j. **sub edx, [var4 + 12] ; Here value of edx = 4000h** **00003ffch**

CLO # 2: **Create basic assembly code using different type of addressing modes in x86 and RISC ISA(s) to solve simple-moderate problems.**

Q3: You have three different size arrays, each having 5 elements. You need to write the code to perform the following operation for each index value (X) ranging from 0 to 4, by using indirect addressing methods. **[Marks: 6]**

arrD[X] = arrD[X] - (arrB[X] + arrW[X])

The data segment has already been defined for you.
While writing code feel free to add any suitable comments to explain your steps.

```
.data
        arrB    BYTE 08h, A4h, A6h, B6h, 10h
        arrW WORD 2 Dup(12h), 2 Dup(0Ah), 01h
        arrDDWORD 5 Dup(FFh)
.code
;start coding from here
```

Question no 3		Points	Rubric
coalMain40 PROC	**;arrD[X] = arrD[X] - (arrB[X] + arrW[X])**		-see next page
mov eax, 0	mov eax, 0		
mov ebx, 0	mov ebx, 0	**Max 1**	Correct Procedure start, exit ENDP and END
mov esi, offset arrB	mov esi, offset arrB	**Max 1**	Correct usage of Mov/add
mov al, [esi]	mov al, [esi+1]	**Max 1**	Indirect Addressing shown
mov esi, offset arrW	mov esi, offset arrW	**Max 1**	Correct use of OFFSET if used
mov bx, [esi]	mov bx, [esi+2]	**Max 1**	Correct Implementation of the equation
add ebx, eax	add ebx, eax		
mov esi, offset arrD	mov esi, offset arrD		

sub [esi], ebx	sub [esi+4], ebx
;correct repetition shown at least once with a suitable comment to justify the concept of redundancy	
exit **coalMain40 ENDP** **END coalMain40**	