**Assignement-1**

**CS-6002 Advance Operating Systems**

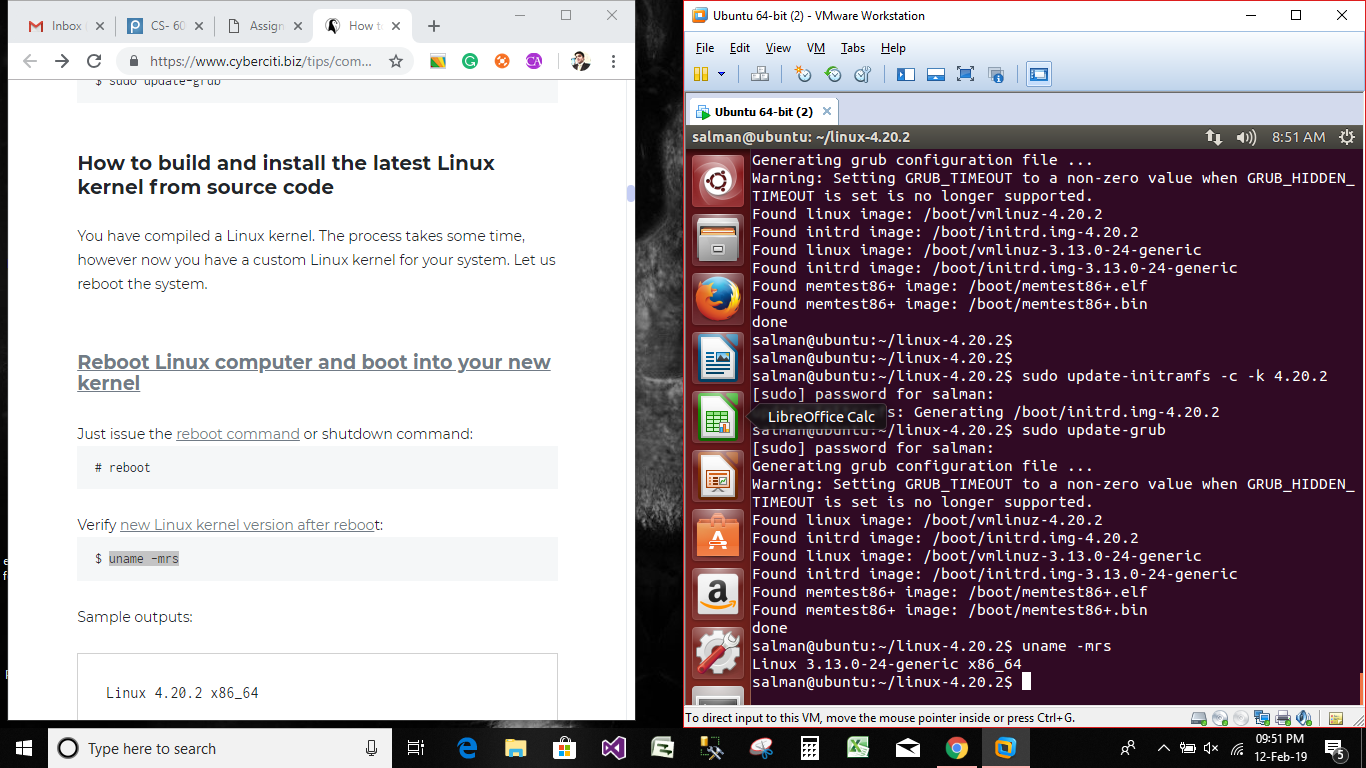
# Due Date: 15th Feburary 2019 (Except part-A)

**Part B)** Download the latest stable kernel source and compile it. You can use following links to guide you <https://www.cyberciti.biz/tips/compiling-linux-kernel-26.html><https://www.linux.com/learn/intro-to-linux/2018/4/how-compile-linux-kernel-0>

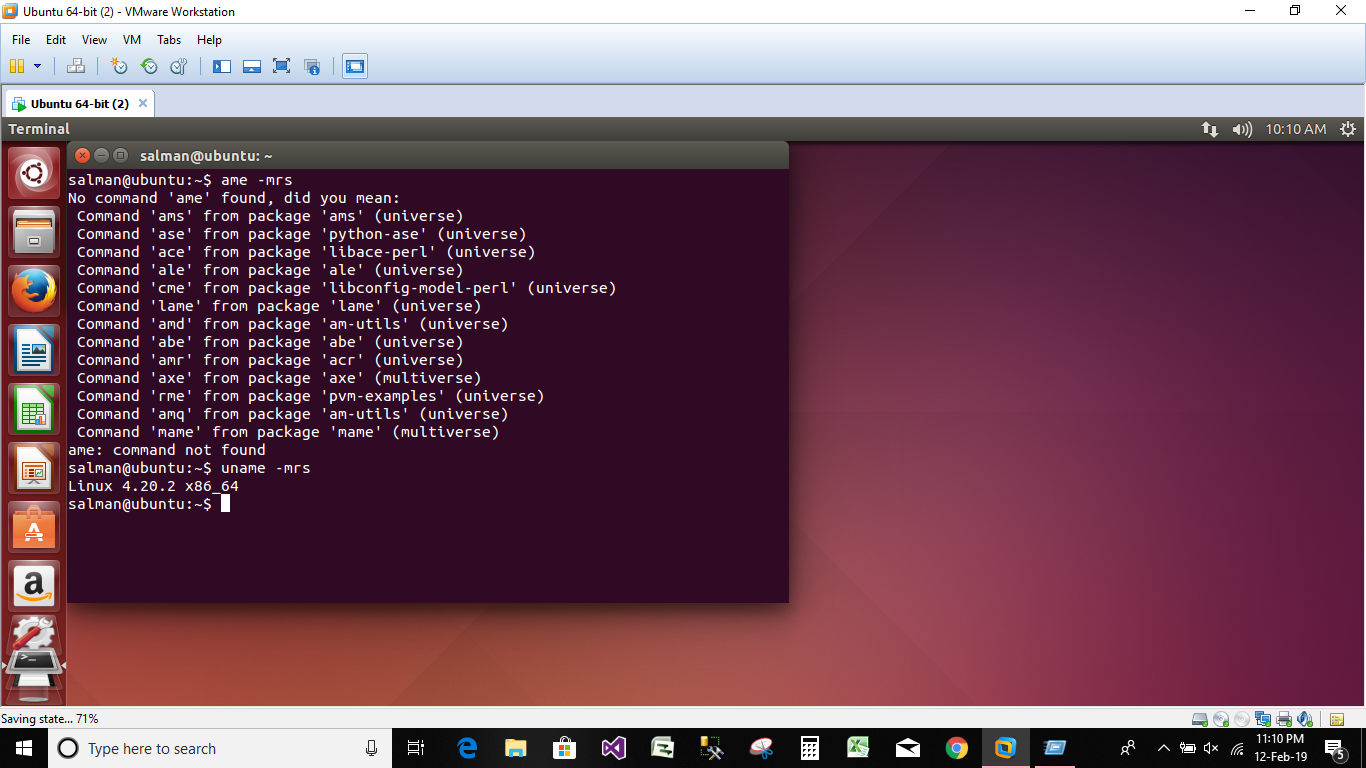
-------------------------------------------------------------------------------------------------------

**Kernal Updated Successfully**

**Kernel status before updating**



**Kernal Status after kernel updated**



**Part C)** UnderstandPointers, Stack & Heap Memory, malloc: When I run the following code on PC.

|  |
| --- |
| #include <stdio.h> #include <stdlib.h> int f(int i,int j, int \*ptr); int main (int argc, char \*argv[]) { int a=2; int b=4;  int \*age = malloc(sizeof(int));  \*age =0;  \*age +=a;  printf("Main Address of local variable a=%u,b=%u and \*age=%u and address of dynamic variable %u\n", (unsigned int)&a,(unsigned int)&b,(unsigned int) &age, (unsigned int) age); f(a,b, age);    return 0;  }  int f(int i,int j, int \*ptr) {  printf("Function f: address of local variable a=%u,b=%u and address of dynamic variable %u\n", (unsigned int) &i,(unsigned int)&j,(unsigned int) ptr);    } |

I get this output:

|  |
| --- |
| ubuntu@ubuntu-VirtualBox:~/code$ ./var0  Main Address of local variable a=3212861652,b=3212861656 and  \*age=3212861660 and address of dynamic variable 158076936  Function f: address of local variable  a=3212861616,b=3212861620 and address of dynamic variable  158076936 |

**Run this code on your computer and explain this output. What these numbers mean, why they are like that and what can we learn from it about the position of the variables.**

**Read this article and try its example on your PC.**

[**https://www.geeksforgeeks.org/memory-layout-of-c-program/**](https://www.geeksforgeeks.org/memory-layout-of-c-program/)

**My Output**



In the output it has shown that the variable storage is dynamic and its changing its address in the memory which can been in output

**Part D)** Compile the code in part-C with “-S” option into assembly and explain its outcome. **Example: compile var.c into var.asm assembly**

# *ubuntu@ubuntu-VirtualBox:~/ gcc –S var.c –o var.asm*

**File Output**

.file "program.c"

.section .rodata

.LC0:

.string "Main Address of local variable a=%u,b=%u and \*age=%u and address of dynamic variable %u\n"

.section .text

.align 4

.proc main

.global main

.type main, @function

main:

.LFB0:

.cfi\_startproc

l.sw -8(r1),r2 # SI store

.cfi\_offset 2, -8

l.addi r2,r1,0 # addsi3

.cfi\_def\_cfa\_register 2

l.sw -4(r1),r9 # SI store

l.addi r1,r1,-36 # allocate frame

.cfi\_offset 9, -4

l.addi r3,r0,2 # move immediate I

l.sw -12(r2),r3 # SI store

l.addi r3,r0,4 # move immediate I

l.sw -16(r2),r3 # SI store

l.addi r3,r0,4 # move immediate I

l.jal malloc # call\_value\_internal

l.nop # nop delay slot

l.ori r3,r11,0 # move reg to reg

l.sw -20(r2),r3 # SI store

l.lwz r3,-20(r2) # SI load

l.addi r4,r0,0 # move immediate I

l.sw 0(r3),r4 # SI store

l.lwz r3,-20(r2) # SI load

l.lwz r4,-20(r2) # SI load

l.lwz r5,0(r4) # SI load

l.lwz r4,-12(r2) # SI load

l.add r4,r5,r4 # addsi3

l.sw 0(r3),r4 # SI store

l.addi r5,r2,-12 # addsi3

l.addi r4,r2,-16 # addsi3

l.addi r3,r2,-20 # addsi3

l.lwz r6,-20(r2) # SI load

l.sw 0(r1),r5 # SI store

l.sw 4(r1),r4 # SI store

l.sw 8(r1),r3 # SI store

l.sw 12(r1),r6 # SI store

l.movhi r3,hi(.LC0) # movsi\_high

l.ori r3,r3,lo(.LC0) # movsi\_lo\_sum

l.jal printf # call\_value\_internal

l.nop # nop delay slot

l.lwz r3,-12(r2) # SI load

l.lwz r4,-16(r2) # SI load

l.lwz r5,-20(r2) # SI load

l.jal f # call\_value\_internal

l.nop # nop delay slot

l.addi r3,r0,0 # move immediate I

l.ori r11,r3,0 # move reg to reg

l.ori r1,r2,0 # deallocate frame

l.lwz r2,-8(r1) # SI load

l.lwz r9,-4(r1) # SI load

l.jr r9 # return\_internal

l.nop # nop delay slot

.cfi\_endproc

.LFE0:

.size main, .-main

.section .rodata

.LC1:

.string "Function f: address of local variable a=%u,b=%u and address\tof dynamic variable %u\n"

.section .text

.align 4

.proc f

.global f

.type f, @function

f:

.LFB1:

.cfi\_startproc

l.sw -8(r1),r2 # SI store

.cfi\_offset 2, -8

l.addi r2,r1,0 # addsi3

.cfi\_def\_cfa\_register 2

l.sw -4(r1),r9 # SI store

l.addi r1,r1,-32 # allocate frame

.cfi\_offset 9, -4

l.sw -12(r2),r3 # SI store

l.sw -16(r2),r4 # SI store

l.sw -20(r2),r5 # SI store

l.addi r5,r2,-12 # addsi3

l.addi r4,r2,-16 # addsi3

l.lwz r3,-20(r2) # SI load

l.sw 0(r1),r5 # SI store

l.sw 4(r1),r4 # SI store

l.sw 8(r1),r3 # SI store

l.movhi r3,hi(.LC1) # movsi\_high

l.ori r3,r3,lo(.LC1) # movsi\_lo\_sum

l.jal printf # call\_value\_internal

l.nop # nop delay slot

l.addi r3,r0,0 # move immediate I

l.ori r11,r3,0 # move reg to reg

l.ori r1,r2,0 # deallocate frame

l.lwz r2,-8(r1) # SI load

l.lwz r9,-4(r1) # SI load

l.jr r9 # return\_internal

l.nop # nop delay slot

.cfi\_endproc

.LFE1:

.size f, .-f

.ident "GCC: (GNU) 4.9.1"

This script generates the assembly code as a file from which it can be seen that the values is been storing in registers and memory and dynamically shifting in memory with the help of registers.