**BPDC, Dubai - First Semester, 2020-2021**

**DEPARTMENT OF CS**

|  |  |  |
| --- | --- | --- |
| **Course No: CS F342**  **Date: Week #05**  **Id No:** | **TUTORIAL 5** | **Course Title: Computer Architecture**  **Name:** |

**1.** Write a MIPS32 program to

a. Compute the length of an asciiz string [String terminated by a null character]

b. Reverse the elements of an asciiz string and store the reversed string in another asciiz string. (no input/output is required)

Answer

The HLL code snippet in C

|  |
| --- |
| str1[10] = “Hello”  for( i=0; str1[i] !=’\0’; i++); // variable ‘i’ will contain the length of the string |

MIPS ALP

|  |
| --- |
| la $a0, str1 #base address of str1 in $a0  li $s0, 0 # i = 0  Loop: add $t1, $s0, $a0 #address of str1[i] in $t1  lbu $t2, 0($t1) #$t2 = str1[i]  beq $t2, $zero, Exit # if str1[i] == ‘\0’, go to Exit  addi $s0, $s0, 1 # i = i + 1  j Loop # go to loop  Exit: # remaining program |

b. After identifying the string length use it to reverse the string.

|  |
| --- |
| la $t0, str1 # base address of str1 in $t0  la $a1, str2 # base address of str2 in $a1  li $s0, 0 # i = 0  ## To find the string length  Loop1: add $t1, $s0, $t0  lbu $t2, 0($t1)  beq $t2, $zero, Loop2  addi $s0, $s0, 1  j Loop1  ## To reverse the string  Loop2: addi $s0, $s0, -1 # i = i -1 points to last char  add $t1, $s0, $t0 # address of str1[i] in $t1  lbu $a0, 0($t1) # $a0 = str1[i]  sb $a0, 0($a1) # store in str2[i]  addi $a1, $a1, 1 # i = I +1  beq $s0, $zero, Exit # if i = 0, go to Exit  j Loop2  Exit: |

**2.** Consider the following C code and answer the following questions w.r.t the same.

|  |  |  |
| --- | --- | --- |
| int main()  {  int i = 20;# i maps to $s0  int j = 30;# j maps to $s1  int x,y,z;  x = func1(i,j);  } | int func1(int x,int y)  {  int a; # a maps to $s1  int b; # b maps to $t0  a = x+y;  b = func2(a,y);  return b;  } | int func2(int x,int y)  {  return x + y;  } |

a. Write the MIPS instruction sequence for calling x = func1(i,j);

b. Write the MIPS instruction sequence occurring at the beginning of func1

c. Write the MIPS instruction sequence for return x + y;

Answer:

a)

- copy the data in $s0, $s1 into argument registers – $a0, $a1

add $a0, $s0, $zero

add $a1, $s1, $zero

jal func1

b)

- Save the $ra because of a nested function call.

- Save $s1 because it is a saved address register

addiu $sp, $sp, -16

sw $ra, 12($sp)

sw $a1, 8($sp)

sw $a0, 4($sp)

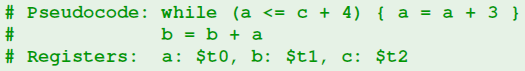
sw $s1, 0($sp)

c)

add $v0, $a0, $a1; #x+y

jr $ra #return to calling procedure

**3.** Consider the following high level language code snippet



a. Convert it into corresponding MALP

b. Assume the starting address of the code is 0, Identify the target address for the branch and jump instructions

**a)**

|  |  |
| --- | --- |
| Address | MALP |
| 0 | addi $t3, $t2, 4 |
| 4 | LOOP: bgt $t0, $t3, END |
| 8 | addi $t0, $t0, 3 |
| 12 | j LOOP |
| 16 | END: add $t1, $t1, $t0 |
| 20 |  |

**b)**

**Branch target address = 16**

**Jump Target Address = 4**

**Branch Target Address = (PC + 4) + offset \* 4**

**= (4+4) + offset \* 4**

**16 = 8 + offset\*4**

**Offset \* 4 = 8**

**Offset(address field) = 2**

**Jump target address = address \* 4**

**4 = address \* 4**

**Address = 1**