Yousef Jarrar

Homework Chapter 2 – 2.4 & 2.27

**2.4)** For the MIPS assembly instructions below, what is the corresponding C statement? Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

\

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| sll | $t0, | $s0, | 2 | # | $t0 | | = | f \* 4 |
| add | $t0, | $s6, | $t0 | # | $t0 | | = | &A[f] |
| sll | $t1, | $s1, | 2 | # | $t1 | | = | g \* 4 |
| add | $t1, | $s7, | $t1 | # | $t1 | | = | &B[g] |
| lw | $s0, | 0($t0) | | # | f | = | A[f] | |
| addi | $t2, | $t0, 4 | |  | | | | |
| lw | $t0, | 0($t2) | |
| add | $t0, | $t0, $s0 | |
| sw | $t0, | 0($t1) | |

* First let’s assume the following information
  + Variable f 🡪 $s0
  + ­Variable g 🡪 $s1
  + Variable h 🡪 $s2
  + Variable i 🡪 $s3
  + Variable j 🡪 $s4

*The starting index of the array A[] is stored in $s6*

*Starting index of the array B[] is stored in $s7*

*The registers $t0, $t1, and $t2 are represented using the variables a, b, and c respectively.*

**The corresponding C statement for each of the instructions given in the question are as follows:**

**#1 Instruction:**sll $t0, $s0, 2

**C-Statement:** a = f << 2;

The instruction sll means shift left logically. The instruction left shifts $s0 by two bits.

In C language, the same operation is done by the operator <<.

**#2 Instruction:**add $t0, $s6, $t0

**C-Statement:** a = A + a;

The instruction add performs the addition operation. The register $s6 contains the base index of array A, which is added with the value stored in register $t0 (represented using a). The result is then assigned back to $t0 (a) itself.

**#3 Instruction:**sll $t1, $s1, 2

**C-Statement:**b = g << 2;

The instruction sll means shift left logically. The instruction left shifts $s1 by two bits.

In C language, the same operation is done by the operator <<.

**#4 Instruction:**add $t1, $s7, $t1

**C-Statement:**b = B + b;

The instruction add performs the addition operation. The register $s7 contains the base index of array B, which is added with the value stored in register $t1 (represented using b). The result is then assigned back to $t1 (b) itself.

*Since, the instruction is to fetch a memory content, an additional operation is required to be performed. For this case an assumption, that one more pointer variable int\* d is already defined, is made.*

**#5 Instruction:**lw $s0, 0($t0)

**C-Statement:**d = a; f = \*d;

• The instruction lw copies data from memory to a register. Since, in C language, to access the content stored at memory, a pointer is required, hence, d is declared as a pointer type.

• The memory address stored in a is assigned to d and the value stored at address pointed by d is assigned to f.

**#6 Instruction:**addi $t2, $t0, 4

**C-Statement:**c = a + 4;

The instruction addi performs addition operation with immediate values such as 4, 10, 100, etcetera. In the given instruction, 4 will be added to the value stored in the register $t0(a) and the result will be assigned to register $t2(c).

*In this instruction also, if a pointer int\* d is already defined.*

**#7 Instruction:**lw $t0, 0($t2)

**C-Statement:**d = c; a = \*d;

• The instruction lw copies data from memory to a register. Since, in C language, to access the content stored at memory, a pointer is required, hence, d is declared as a pointer type.

• The memory address stored in c ($t2) is assigned to d and the value stored at address pointed by d is assigned to a ($t0).

**# 8 Instruction:**add $t0, $t0, $s0

**C-Statement:**a = a + f;

The instruction add performs the addition operation. The values of registers $s0(f) and $t0(a) are added and then stored in register $t0(a).

*For this instruction, if a pointer variable int\* d is already defined.*

**#9 Instruction:**sw $t0, 0($t1)

**C-Statement:** d = b; \*d = a;

• The instruction sw is used to store the content from a register to a memory location. In this instruction, the content stored in $t0 is transferred to memory location pointed by $t1.

• The first C-statement, copies the address to d.

• Next C-statement, copies the content of a ($t0) to the location pointed by the pointer variable d

*Since, lw and sw operations are used to transfer data between memory and registers, these operations can’t be completed in a single statement. Hence, two statements are used for each of these instructions.*

**2.27)** Translate the following C code to MIPS assembly code. Use a minimum number of instructions. Assume that the values of a, b, i, and j are in registers $s0, $s1, $t0, and $t1, respectively. Also, assume that register $s2 holds the base address of the array D.

for(i=0; i<a; i++)

for(j=0; j<b; j++) D[4\*j] = i + j;

**MIPS Instruction Code:**

LOOP1:

beq $t0, $s0, EXIT

add $t0, $zero, $zero # init the value of i in t0

add $t4, $s2, $zero # init value of t4 with s2

LOOP2:

beq $t1, $s1, LOOP1

add $t3, $t0, $t1

sw $t3, 0($t4)

addi $t4, $t4, 4

addi $t1, $t1, 1

slt $s3, $t1, $s1

bne $s3, $zero, LOOP2

addi $t0, $t0, 1

j LOOP1

EXIT:

* The LOOP1 acts as the outer for loop. The purpose of this loop is to initialize the content of $t1 (variable j) register.
* The LOOP1 also initializes the register $t4 with the base address of array D stored in $s2.
* In the LOOP2, perform the operation done by the inner for loop.
* In the LOOP2, compute the result of i + j which is equal to $t0 + $t1.
* Increment the value of address stored in $t4 by 4.
* Increment the $t1 by 1.
* If the condition of inner loop is not matched, jump to the LOOP2.
* Otherwise, increment the $t0 by 1 and jump to LOOP1.