Assignment 2

Zuo Yiyang No: 1220008661

Q1: The following problems deal with translating from C to MIPS. 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. Given the C statement f = -g – A[6];

(1) For the C statement above, what is the corresponding MIPS assembly code?

(2) For the C statement above, how many different registers are needed to carry out the C statement?

Solution:

(1) We can divide the statement into a few parts to deal with them respectively.

Step 1: f = A[6] ⇒ lw $s0, 24($s6)

Step 2: f = -A[6] ⇒ sub $s0, zero, $s0

Step 3: f = -A[6] – g ⇒ sub $s0, $s0, $s1

Thus, the corresponding MIPS assembly code is the three listed above.

(2) Since we use registers $s0, $s6 and $s1, three different registers are needed to carry out the C statement.

Q2: The following problems deal with translating from MIPS to C. 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. Given the MIPS assembly codes: sll $s2, $s4, 4, add $s0, $s2, $s3, add $s0, $s0, $s1

(1) For the MIPS assembly instructions above, what is the corresponding C statement?

(2) For the MIPS assembly instructions above, rewrite the assembly code to minimize the number if MIPS instructions (if possible) needed to carry out the same function.

(3) How many registers are needed to carry out the MIPS assembly as written above? If you could rewrite the code above, what is the minimal number of registers needed?

Solution:
(1) From the given MIPS codes, we can obtain the corresponding C statement:
Step 1: sll $s2, $s4, 4 $\Rightarrow$ h = 8*j;
Step 2: add $s0, $s2, $s3 $\Rightarrow$ f = h + i;
Step 3: add $s0, $s0, $s1 $\Rightarrow$ f = f + g;
Thus, we can obtain: f = 8*j + i + g;

(2) Noting the statement obtained above, including addition and multiplication, which can be implemented by single(cannot be divided) MIPS codes, thus the given MIPS codes are the most optimized solution.

(3) From the analysis in the previous subproblem, we have known that the MIPS codes cannot be divided again, thus 5 registers as written, and 5 minimally.

Q3: Show the single MIPS instruction or minimal sequence of instructions for this C statement: b = 25 | a; Assume that a corresponds to register $t0 and b corresponds to register $t1.
Solution:
Since there is an immediate number in the given statement, the corresponding MIPS code is: ori $t1, 25, $t0.

Q4: Convert the following MIPS instructions to the corresponding machine codes. Assume the instructions are stored in memory with the address listed left.

```
start: beq $s0, $a0, next          #memory address [0x00000000]

       sub $v0, $a0, $s0           #memory address [0x00000004]

next: addi $s0, $s0, 1             #memory address [0x00000008]

       j   start                   #memory address [0x0000000C]
```

Solution:

1st line(I-format): We have obtained op = beq = 000100, rs = s0 = 16 = 10000, rt = a0 = 4 = 00100, and immediate, and we need to compute the immediate by this conditional branch.

$$\text{target PC} = \text{PC} + 4 + \text{imm} \cdot 4 \Rightarrow \text{imm} = (\text{target PC} - \text{PC})/4 - 1$$

Thus, immediate = (0x00000008 – 0x00000000)/4 – 1 = 0x00000001 = 0000000000000001

Thus, the machine code is 0001 0010 0000 0100 0000 0000 0000 0001 = 1 2 0 4 0 0 0 1

2nd line(r-format): op = sub = 000000, rs = a0 = 4 = 00100, rt = s0 =16 = 10000, rd = v0 = 2 = 00010, shamt = 00000, funct = 100010

Thus, the machine format = 0000 0000 1001 0000 0001 0000 0010 0010 = 0 0 9 0 1 0 2 2

3rd line(I-format): op = addi = 001000, rs = s0 = 16 = 10000, rt = s0 = 16 = 10000, immediate = 1 = 0000000000000001

Thus, the machine code is 0010 0010 0001 0000 0000 0000 0000 0001 = 2 2 1 0 0 0 0 1

4th line(J-format): We have obtained op = j = 000010, to determine the value after the opcode with 26 bits, we need to find the address of jump target first. The address of start is: 0000 00000000000000000000000000 00. Since the address of target PC = (upper 4 bits of (PC + 4): 26 bit address) << 2, delete the last two bits and the first 4 bits, leaving the 26 bits. Thus, j start $\Rightarrow$ 000010 00000000000000000000000000, and

the machine code is 0000 1000 0000 0000 0000 0000 0000 0000 = 0 8 0 0 0 0 0 0

Q5: Write a procedure, *find_k*, in MIPS assembly language. The procedure should take a single argument that is a pointer to a null-terminated string in register $a0. The *find_k* procedure should locate the first 'k' character in the string and return its address in register $v0. If there is no 'k' in the string, then *find_k* should return a pointer to the null character at the end of the string. For example, if the argument to *find_k* points to the string "hijklmn", then the return value will be a pointer to the fourth character of the string.

Solution:

Initially, a null string is stored in register a0, and we need to give a string to the register to finish our procedure. Thus, we need to define the following registers to complete the goal.

t0 → register to store the 'target' character 'k'

t1 → a temporary register to store the current character in the given string

t2 → a temporary register for the special case that the given string is null

t3 → a temporary register for the case that the 'target' is found

Thus, we can obtain the following MIPS assembly codes to illustrate the required procedure.

```
find_k: addi $t0, $zero, 107 /*The ASCII code of k is 107*/
loop: lw $t1, 0($a0) /*load the first string to the register*/
        seq $t2, $t1, $zero /*if($t1 == 0), $t2 = 1, indicating that the given string is null*/
        bne $t2, $zero, finish
        seq $t3, $t1, $t0 /*if the current character is 'target'*/
        bne $t3, $zero, finish /*If the 'target' is found, finish the procedure*/
        addi $a0, $a0, 4 /*else continue to search*/
        j loop
finish: move $v0, $a0
         jr $ra
```