CSE3401 Computer Organization

Fall 2023 – HW#1

Due: 01.11.2023

Q1) B. ASSUME YOU HAVE $t0=[0xCAFEBABE], HOW CAN YOU CHANGE IT TO

[0xBABACAF0]

and

[0x03CA1E00]

Using logic and shift operations.

Q1.1 Answer

```
.data
theword: .word 0xcafebabe
.text
 lw $t0,theword
```

CSE3401 Computer Organization

Fall 2023 – HW#1

Due: 01.11.2023

```
srl $t1,$t0,16                  #0000CAFE
andi   $t1,$t1,0x0000fff0   #0000CAF0
sll $t2,$t0,16                  #BABE0000
andi $t2,$t2,0xfff00000  #BAB00000

 or $t1,$t1,$t2                 #BAB0CAF0

andi  $t0,$t0,0x0f000000  #0A000000
srl $t3,$t0,8                   #000A0000
 or $t0,$t3,$t1                 #BABACAF0
```

.data section:

A label called theword was supplied, and the value 0xcafebabe (in base 16 since it begins with 0x) was saved in memory under this label.

.text section:

We loaded the value beneath the theword tag from memory into the 0xcafebabe $t0 register using the lw $t0, theword instruction.

We moved the value in register $t0 by 16 bits to the right and recorded the outcome to register $t1 using the srl $t1, $t0, 16 instruction. Now for the "0xcafe" portion.

The $t1 register was treated to a bitwise AND operation with 0x0000fff0 using the andi $t1, $t1, 0x0000fff0 instruction, and the result was written to the $t1 register. Thus, the "0xcaf0" portion was obtained.

We wrote the outcome to register $t2 after shifting the value in register $t0 by 16 bits to the left using the sll $t2, $t0, 16 command. This is where the "0xbabe0000" portion comes from.

The $t2 register was treated to a bitwise AND operation with 0xfff00000 using the andi $t2, $t2, 0xfff00000 command, and the result was written to the $t2 register. Thus, the "0xbab00000" portion is what we have.

Using the or $t1, $t1, $t2 command, we performed a bitwise OR operation on the registers $t1 and $t2, writing the outcome to the $t1 register. This indicates that the "0xbab0caf0" component has been produced.

The $t0 register was treated to a bitwise AND operation with 0x0f000000 using the andi $t0, $t0, 0x0f000000 command, and the result was written to the $t0 register. Thus, the "0x0a000000" portion was obtained.

We moved the value in register $t0 by 8 bits to the right and wrote the outcome to register $t3 using the srl $t3, $t0, 8 command. Thus, the "0x000a0000" portion was obtained.

We performed a bitwise OR operation on the $t3 and $t1 registers using the or $t0, $t3, $t1 command, and then we wrote the outcome to the $t0 register. In this manner, the "0xbabacaf0" portion was produced.

We generated the value "0xbabacaf0" in the $t0 register as a result of these activities.

Q1.2 Answer

```
.data
theword: .word 0xcafebabe

.text
 lw $t0, theword                #load $to=0xcafebabe
```

CSE3401 Computer Organization

Fall 2023 – HW#1

Due: 01.11.2023

```
andi $t0, $t0, 0x00ff0f00  # $t0 & 0x00ff0f00  make and operation
li $t1, 0x03ca1e00         #load value 0x03ca1e00 to $t1
or $t0, $t0, $t1           # or the values $t0 and $t1
```

We bitwise ANDed the value in register $t0 to 0x00ff0f00 using the andi $t0, $t0, 0x00ff0f00 command, and then wrote the outcome to register $t0. This procedure resets some bits of $t0 and preserves others.
We used the instruction li $t1, 0x03ca1e00 to load the value 0x03ca1e00 into register $t1.
The $t0 and $t1 registers are subjected to a bitwise OR operation with the or $t0, $t0, $t1 instruction, and the result is written to the $t0 register. We did this by combining specific bits of $t0 with the matching bits of $t1.
Consequently, this code adds 0x03ca1e00 to the value that was initially 0xcafebabe after filtering specific portions of $t0 with 0x00ff0f00. The result is then written within $t0. Hence, $t0's final value was 0x03ca1e00.

Q2) WHAT MIPS INSTRUCTION IS REPRESENTED BY THE FOLLOWING HEX

A. 0x01090010

Q2 Answer

000000	01000	01001	00000	00000	010000
6 bits op	5 bits rs	5 bits rt	5 bits rd	5 bits shamt	6 bit funct

We arranged the given expression according to its bits, we know that the first 6 bits we get allow us to find out what format the MIPS is in. According to this example, the first 6 bits consist of 0s, which means that our MIPS has the R type format. We placed the binaries I created according to the R type format, so we placed their binary equivalents according to op, rs, rt, rd, shamt and func locations. We found the numerical equivalents for rs, rd, rt, shamt and func. rs=8, rt=9, rd=0, shamt=0, funct=16. We looked at the operation to be performed according to the information given in MIPS addressing for 32 bits immediates and addresses, and we saw that it was mfhi and expressed it as follows:

mfhi $t0,$t1,$zero

Q3) Convert the following java code into MIPS

```
int i = 23;
while (i>5)
    i--;
```

CSE3401 Computer Organization

Fall 2023 – HW#1

Due: 01.11.2023

Q3 Answer

```
.data
i: .word 23
.text
.globl main
main:
   lw $t0, i
   li $t1, 5

loop:
   slt $t2, $t1, $t0
   beq $t2, $zero, end_loop

   sub $t0, $t0, 1
   sw $t0, i            j loop

end_loop:
   li $v0, 10
   syscall
```

In converting a simple loop structure from Java to MIPS assembly language, we first assign the initial value of 23 to the variable i. In MIPS, this is stored as a word under the label i in the .data section. In the main section of the program (main), this value is loaded into the $t0 register. The while loop from Java is replicated in MIPS with a loop denoted by a label (loop). Within this loop, $t0 is compared with 5 ($t1), and if $t0 is greater, it is decremented by one, and the updated value is written back to the location of i. This process continues until $t0 is equal to or less than 5. When the loop concludes, the program is terminated with a system call (syscall). This logic effectively models the core principles of the Java while loop using MIPS assembly language commands.

MELİKE TEPELİ 20COMP1012