Prof. Li-Yan Yuan

CMPUT 229 COMPUTER ORGANIZATION AND ARCHITECTURE (I)

Solutions to Midterm Examination

October 27, 2006

1. Name the three different formats of MIPS instructions, and give an example of each type. [5]

   Solution:

   (a) R-type, or register type; for example: ADD $t2, $t1, $t0

   (b) I-type, or immediate type; for example, ADDI $t2, $t1, 10

   (c) J-type, or jump type; for example, J somewhere

2. The program counter in many microcomputer architectures is usually incremented by one while the MIPS program counter is incremented by four. Why is the amount different? [5]

   Solution: This is because MIPS is byte-addressable, and each instruction is 4 bytes long.

3. A machine has 12 bit instructions with two different formats, that is, one-address instructions and two-address instructions. Each address is 4 bits long and all instructions consist of only an opcode and the address(es). Assuming that the instruction encoding space is completely utilized and both kinds of instructions exist, what is the maximum number of two-address instructions? Justify your answer briefly. [5]

   Solution: A two-address instructions uses 8 bits for its two addresses and thus has only 4 bits left for its opcode. This gives $2^4 = 16$ possible operation codes.

   To accommodate the need for one-address instructions, at least one operation code must be allocated for one-address instructions. Therefore, the maximum number of two-address instructions is 15.

4. Consider the following MIPS instructions. Show the contents of each relevant register in each case after the instruction is executed, assuming that $t0 contains 0x0435BF33 and $t1 contains 0xFC0A4024 at the start of each instruction. [5]

   (a) srav $t2, $t0, 5

   (b) xor $t2, $t1, $t0

   Solution: In both cases, $t2 is the only (concerned) register whose value has been changed. The chnaged value of $t2 is given below

   (a) $t2: 0021ADF9

   (b) $t2: F83FFF17

5. . Write a program in MIPS assembly language which performs the following steps: [10]

   (a) prompts for the user to enter TWO integers; (to display a simple string such as “enter two integers”, and then to read two intergers using syscall)

   (b) if the second number is not 0 then print out the quotient of dividing the first number by the second one (ignore the remainder);

   (c) otherwise, print out ”the second number shall not be 0”.

   Solution: A sample program for the question is given below.

```
##
## Soltion to Qestion 5 on Midterm, October 27, 2006
##
##-----------------------------------------------
#               text segment
##-----------------------------------------------
        .text
main:   # display the prompt
        la $a0, prompt
        li $v0, 4
        syscall

        # read two integers into $t0 and $t1
        li $v0, 5
        syscall
        move $t0, $v0

        li $v0, 5
        syscall
        move $t1, $v0

        # check for zero divisor
        beqz $t1, zero_divisor

        # calcuate and then store the quotient in $v0
        div $a0, $t0, $t1

        # print out the quotient
        li $v0, 1
        syscall

        j exit

zero_divisor:
        la $a0, errormsg
        li $v0, 4
        syscall

exit: # exit
        li $v0, 10
        syscall

#-----------------------------------------------
#           data segment                       |
#-----------------------------------------------
        .data
prompt: .asciiz "enter two integers\n"
errormsg:
        .asciiz "the second number shall not be zero\n"

##
## end of file midt_q5.s
```

6. The following C program calls a library function `int atoi(char *)` to print an integer that is converted from the initial portion of a string.

```
/*
 *  to demo how to use the library function  int atoi(char *)
 */
#include <stdlib.h>
char string[]="123abcd";
int main(void) {
  printf("%d\n", atoi(&string[0]));
}
```

A simplified specification of the function `ATOI` is given below.

```
ATOI(3)   Linux Programmers Manual
NAME
       atoi - convert a string to an integer
SYNOPSIS
       #include <stdlib.h>
       int atoi(const char *nptr);
DESCRIPTION
       The atoi() function converts the initial part of the string in nptr
       to an integer (decimal number) value in the obvious manner,
       stopping at the first character which is not a valid digit.
RETURN VALUE
       The converted value.
```

For simplicity, assume that valid digits are '0', '1', ..., '9', with respective ASCII integer values 48, 49, ..., 57.

Translate the given C program, together with the implementation of the function `ATOI`, into a MIPS program.

You must follow the CMPUT 229 Conventions in your programming. [15]

```
## Soltion to Qestion 6 on Midterm, October 27, 2006
#-----------------------------------------------
#           text segment                       |
#-----------------------------------------------
        .text
main:   # call ATOI function, using $a0 to pass the address of the string
        la $a0, string
        jal atoi

        # print the return value from ATOI
        move $a0, $v0
        li $v0, 1
        syscall

        exit:   # exit
        li $v0, 10
        syscall
```

```
# ATOI function
        # to conver the initial portion of the given string an integer
        # $a0:  taking the address of the given string
        # $v0:  to return the converted integer value
        # $t0:  the address of the current char
        # $t1:  the ASCII value of the current char

atoi:   # to set up the stack frame for $ra
        sub $sp, $sp,4
        sw  $ra, 0($sp)

        # move the input to the address of the current char
        move $t0, $a0

        # to initialize the result to 0
        li  $v0, 0

next:   # fetch the current char
        lb $t1, ($t0)

        #  check if the current char is a valid digit
        blt $t1, 48, endloop
        bgt $t1, 57, endloop

        #  to update result $v0 = v0*10 + $t1 - 48
        mul $v0, $v0, 10
        add $v0, $v0, $t1
        sub $v0, $v0, 48

        # to advance the address to point to the next char
        add $t0, $t0, 1

        # continue for the next char
        b next

endloop:
        # to pop the return address
        lw $ra, 0($sp)
        add $sp, $sp, 4

        # to terminate the function
        jr $ra
#---------------------------------------------
#           data segment                      |
#---------------------------------------------
        .data
string: .ascii "1234"

## end of file midt_q5.s
```