CS61c-Midterm 1, Professor Fateman: 9/29/98

Your family name _____ Your given name_____
Your Student ID number _____ login: cs61c_____

Please circle the last two letters of your login name.

b c d e f g h i j k l m n a o p q r s t u v w x y z
b c d e f g h i j k l m n a o p q r s t u v w x y z

The reason we asked this was so we would have some recourse if we could not read your
handwriting. A surprising number of people left this out, including people whose login was unclear to us, confusing g and q, u and v, l and e, a and o.

Discussion section meeting time_____ TA's name _____

Look at the edge of your seat. Write your row and number. Your row number may not be visible
from where you sit, so we will help you later. Row_____ Seat _____

This booklet contains 6 numbered pages including the cover page, plus 3 pages of excerpts from
appendix B and C of Goodman and Miller. Put all answers on these pages, please; don't hand in
stray pieces of paper. The exam contains 7 substantive questions, plus question 0 and the box
immediately below.

I certify that my answers to this exam are all my own work, and that I have not discussed the exam questions or answers with anyone prior to taking this exam. If I am taking this exam early, I certify that I shall not discuss the exam questions or answers with anyone until after the scheduled exam time.
Signature _____

Question	Max Points	Your Points
0	1	
1	17	
2	5	
3	6	
4	6	
5	5	
6	5	
7	15	
8	20	
Total	80	

Question 0 (1 point): Fill out the front page correctly and put your login correctly at the top of
each of the following pages..

Question 1 (20 points): These questions pertain to the mostly meaningless "program" on the
next page.

a. Place a check in the appropriate column to identify which of the MAL instructions used below
must be changed to more than one machine instruction (that is, requires more than one instruction when written in pure TAL). **And write out their translation into TAL in the space at the bottom of the page!**

b. Certain of the commands are not legal assembly language. Identify them with a checkmark as
well.

c. At a number of points in the execution of this program, the value in register 4 changes. Place a
checkmark in the appropriate column when, after the indicated instruction completes, register 4
has the number 5 in it.
You may assume that the machine you are executing the instructions on is an HP (big-endian)
machine.

```
            .data
word5:      .word 5
abc5:       .byte ’a’, ’b’,’c’,5
zero:       .word 0
      .text
__start:    #                    (1a)                (1b)        (1c)
      # The main routine      Check if 2 ops     Check if it      Check
if
      #                       in TAL             is illegal
[$4]=5
main:       lui $4, 0   1     -----------------|----------------|------
---------
      ori $4, 5         2     -----------------|----------------|------
---------
      lui $4, 0         3     -----------------|----------------|------
---------
      li $4, 0x50005    4     -----------------|----------------|------
---------
      la $4, word5      5     -----------------|----------------|------
---------
      lw $4, word5      6     -----------------|----------------|------
---------
      addi $4,$0,0      7     -----------------|----------------|------
```

```
---------
      ori $4, $4,0x101  8     ----------------|---------------|------
---------
      andi $4, $4,0x0   9     ----------------|---------------|------
---------
      ori $4, $4, 0x5   10    ----------------|---------------|------
---------
      lw $4, main+4     11    ----------------|---------------|------
---------
      sll $4, $4,16     12    ----------------|---------------|------
---------
      srl $4, $4,16     13    ----------------|---------------|------
---------
      addi $4, $0,4     14    ----------------|---------------|------
---------
      addi $4, $0,1     15    ----------------|---------------|------
---------
      lw $4, zero       16    ----------------|---------------|------
---------
      addi $4, 4(abc5)  17    ----------------|---------------|------
---------
      add $4, abc5+3(0) 18    ----------------|---------------|------
---------
      lw $4, word5($0)  19    ----------------|---------------|------
---------
      sw $4, word5      20    ----------------|---------------|------
---------
      lb $4, abc5       21    ----------------|---------------|------
---------
      lb $4, abc5+3     22    ----------------|---------------|------
---------
      subi $4,$0,-5     23    ----------------|---------------|------
---------
done
```

Question 2 (5 points). Here is a 2-instruction TAL program that is intended to load into register $4 the address in memory of its first instruction, the one at the location "here".

```
here: ????
step2: addi $4, $31,-4
```

What instruction would you put in place of ????

Question 3 (6 points). Write down the number 2048(base 10) in hexadecimal and binary.

Assuming you have a 2’s complement representation, what is -2048 as a 16-bit binary number?

Now write that number as a positive hexadecimal quantity.

Question 4 (6 points). What do the following expressions evaluate to in C? (Show your work)
((1 & 4) | 3) & (~5) =
((1 && 4) || 3) && (!5) =

Question 5 (5 points). As we have discussed in lecture, one strategy in setting up register usage is to have one register, a global pointer, GP point to the beginning of the data segment of
your program. That way it is easy for all programs to refer to data like error message strings
with simple offsets from that base register.
Instead of setting GP to the beginning(numerically lowest) address in the .data segment, say
0x10000000, it may be useful to set it somewhere above
the lowest address in use, say to 0x10008000. Why?

Question 6 (5 points). On the MIPS, architecture, the only addresses that one can branch to
are on full-word boundaries (have 2 0s at the end), and therefore all branch instructions always
use the 16-bit immediate as an 18 bit signed field. Since one can only load words from a full-word
location, why aren't ALL the offsets in all the load and store instructions also multiplied by 4?

Question 7 (15 points). Here is a simple program to compute a function of one argument. It is
based on a program you have seen in the lab, but this time it is not recursive.
a. See if you can make it smaller by eliminating (crossing out) unnecessary lines of assembler.

```
__start:
            li    $a0,4       #f(4)
            jal   f
            done

f:
            sub   $sp,$sp,8         # adjust the stack for 2 items
            sw    $ra,4($sp)        # save the return address
            sw    $a0,0($sp)        # save the argument n
            addi  $v0,$zero,1       # initialize answer

loop:       slt   $t0,$a0,2         # test for n < 2
            beq   $t0,$zero,L1      # if n>1, go to L1
            addi $sp,$sp,8          # pop off 2 items off stack
```

```
            jr    $ra               #return to after jal

L1:         mul   $v0,$a0,$v0
            sub   $a0,$a0,1         # n >=1; argument gets (n-1)
            j     loop
```

b. There is no subi instruction on MIPS. Make believe you are the MAL assembler and convert
subi $a0,$a0,1 into one or more actual machine instructions (express these
in TAL):

c. Assume that the location of the instruction beq $t0,$zero,L1
is 0x00400028.
That instruction includes a field with a 16-bit displacement representing the distance K to branch to get to L1. (In hexadecimal), what is the value of K? (explain) .

e. Finally, what does f(4) return?

Question 8 (20 points). Translate the following C procedure into MIPS assembler. Try to keep
it brief! Use the usual conventions which assume that the arguments to procedures are kept in
$a0-$a4 (regs 4-7) , temporary variables (preserved across call) are kept in $s0-$s7 (regs 16-23) .
These can be used for internal computation but must be preserved by the callee across procedure
calls. The return value is $v0 (reg 2).

```
/* Question 8 */
int foo(int x, int *a){
     int z;
     if (x < 0) z = 0;
     else z = bar (a[x]);
     return z;
}
```