University of California, Berkeley
College of Engineering
Computer Science Division — EECS

Spring 2001 John Kubiatowicz

Homework Quiz (HW #2)
SOLUTIONS

February 6, 2001
CS152 Computer Architecture and Engineering

This quiz covers one of the problems from homework #2.
Good Luck!

Your Name:	
SID Number:	
Discussion Section:	

Total:	

General ASCII Number Conversions

In homework problem 3.25, you were asked to write a MIPS assembly language program to convert an integer input value into the ASCII string which is its base-10 representation. For this quiz, construct a similar program that will convert an integer into a string of *any* base (from 2 to 16). The specifications are as follows:

- The first argument (a0) contains the *unsigned* input value
- The Second argument (a1) contains the base and will be such that: $2 \leq a1 \leq 16$ (you can assume that this is true)
- The Third argument (a2) will be a pointer to a buffer in memory for the result. Resulting strings should be *null terminated* (i.e. the end of the string is 0).
- On exit, v0 will contain the length of the resulting string
- This procedure (call it "intoa") must adhere to all MIPS conventions

Examples:

If the input value (a0) is 14, the base (a1) is 2, then the output string (pointed at by a2) should be "1110" (followed by the null character) and the result register (v0) should contain 4.

If the input value (a0) is 182, the base (a1) is 16, then the output string (pointed at by a2) should be "B6" (followed by the null character) and the result register (v0) should contain 2.

If the input value (a0) is 153, the base (a1) is 10, then the output string (pointed at by a2) should be "153" (followed by the null character) and the result register (v0) should contain 3.

The important portion of the ASCII character table is as follows (Note that values are in HEX notation):

Character	ASCII Value
"0"	0x30
"1"	0x31
"2"	0x32
"3"	0x33
"4"	0x34
"5"	0x35
"6"	0x36
"7"	0x37
"8"	0x38
"9"	0x39

Character	ASCII Value
"A"	0x41
"B"	0x42
"C"	0x43
"D"	0x44
"E"	0x45
"F"	0x46

We have included the MIPS instruction set as an attachment. Write this routine in as few instructions as possible on the following page. Assume the virtual MIPS machine (no branch delay slots).

[This page is for your answer]

This is a straightforward extension of the homework exercise. This one tricky part about this solution is that we need to reverse the result, since we generate the digits in reverse order. Consequently, we will write them out to the memory buffer, the reverse them in place. Note that this is set up so that the string is always at least of length 1 (since we always go through the loop at least once). This really means that we suppress leading zeros except if the value is 0.

```
intoa: add    $v0, $r0, $r0      ; Initialize length to zero
       add    $t0, $a0, $r0      ; Remainder
       add    $t1, $a2, $r0      ; Pointer to destination buffer
loop:  divu   $t0, $a1           ; Compute next digit
       mflo   $t0                ; Save quotient (remaining digits)
       mfhi   $t2                ; Get remainder => next digit
       sltui  $t3,$t2,10         ; Less than 10?
       bne    $t3,save           ; Yes
         addi $t2,$t2,0x30       ; Add in ASCII for '0' (in delay slot)
       addi   $t2,$t2,7          ; Not less than 10.  Adjust so 10=>0x41
save:  st     $t2,0($t1)         ; Save next digit
       addi   $v0,$v0,1          ; One more digit!
       bne    $t0, loop          ; More digits to go
         addi $t1,$t1,1          ; Increment to next digit location
       st     $r0,0($t1)         ; Make sure to null-terminate string

       addi   $t0,$a2,$r0        ; Head of string
       addi   $t1,$t1,-1         ; Back up to previous char
       ld     $t4,0($t0)         ; swap two values
rev:   ld     $t5,0($t1)
       st     $t4,0($t1)
       st     $t5,0($t0)
       addi   $t0,$t0,1
       addi   $t1,$t1,-1
       slt    $t3,$t1,$t0        ; Oops.  Pointers passed each other
       beq    $t3, rev
         ld   $t4,0($t0)         ; Filled delay slot
exit:  jr     $ra                ; All done
```