Homework Quiz (HW #2)
SOLUTIONS
February 12, 2003
CS152 Computer Architecture and Engineering

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

<table>
<thead>
<tr>
<th>Your Name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID Number:</td>
</tr>
<tr>
<td>Discussion Section:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total:</th>
</tr>
</thead>
</table>
General Base Conversions

In homework problem 3.22, you were asked to write a MIPS assembly language program to convert an ASCII decimal string to an integer. For this quiz, construct a similar program which can convert a string of any base (from 2 to 16) to an integer. The specifications are as follows:

- The first argument (a0) contains a pointer to a null-terminated string
- The Second argument (a1) contains the base and will be such that: \(2 \leq a1 \leq 16\) (you can assume that this is true)
- On exit, v0 will contain the result.
- This procedure (call it “convert”) must adhere to all MIPS conventions

If any character in the string (before the final null character) is not a legal character for the specified base, then the result should be –1. Also, a zero-length string (no characters at all) should return –1. Ignore overflow errors.

Examples:
- If the input string is “B6” and the base is 16, the result register (v0) should contain 182
- If the input string is “B6” and the base is 12, the result register, should contain 138
- If the input string is “B6” and the base is 10, the result register, should contain –1
- If the input string is “” (i.e. first character is null), the result register should contain –1
- If the input string is “10110” and the base is 2, the result register, should contain 22.

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

<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII Value</th>
<th>Character</th>
<th>ASCII Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“0”</td>
<td>0x30</td>
<td>“A”</td>
<td>0x41</td>
</tr>
<tr>
<td>“1”</td>
<td>0x31</td>
<td>“B”</td>
<td>0x42</td>
</tr>
<tr>
<td>“2”</td>
<td>0x32</td>
<td>“C”</td>
<td>0x43</td>
</tr>
<tr>
<td>“3”</td>
<td>0x33</td>
<td>“D”</td>
<td>0x44</td>
</tr>
<tr>
<td>“4”</td>
<td>0x34</td>
<td>“E”</td>
<td>0x45</td>
</tr>
<tr>
<td>“5”</td>
<td>0x35</td>
<td>“F”</td>
<td>0x46</td>
</tr>
<tr>
<td>“6”</td>
<td>0x36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“7”</td>
<td>0x37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“8”</td>
<td>0x38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“9”</td>
<td>0x39</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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).
## HOMEWORK QUIZ FOR HW#2
## Henry Lam 2/22/2003
##
## Your solutions may vary and could be more
elegant than this one.

convert:
```assembly
addu $v0, $0, $0  # initialize output
addiu $t0, $0, 0  # offset to address
addu $t2, $0, $0  # zero out $t2
addu $t3, $0, $0  # zero out $t3
addiu $t9, $0, 0x40  # $t9 = 0x40
lbu $t1, 0($a0)  # check for empty string
beq $t1, $0, error  # string invalid
```

getchar:
```assembly
addu $t1, $t0, $a0  # memory address
lbu $t1, 0($t1)  # t1 = char
addiu $t0, $t0, 1  # increment the offset
beq $t1, $0, done  # reach null pointer = done
slt $t2, $t1, 0x30  # check for lower boundary
bne $t2, $0, error  # char invalid
slt $t2, $t1, 0x47  # check for upper boundary
beq $t2, $0, error  # char invalid
slt $t3, $t1, 0x3A  # char is 0-9
bne $t3, $0, dec  # go to decimal conversion
slt $t3, $t9, $t1  # char is alpha
bne $t3, $0, alpha  # go to alpha conversion
j error  # char is 0x3A - 0x40 = invalid
dec:
addu $t1, $t1, -48  # subtract 48 to convert to integer
j charready  # character is ready
```

alpha:
```assembly
addiu $t1, $t1, -55  # subtract 55 to convert to integer
j charready  # character is ready
```

charready:
```assembly
multu $v0, $a1  # multiply output by base
mflo $v0  # move the result to $v0
# assumes answer > 32 bits
addu $v0, $v0, $t1  # add integer to old result
addu $t2, $0, $0  # reset $t2
addu $t3, $0, $0  # reset $t3
j getchar  # get next character
done:
jr $ra  # return
error:
addiu $v0, $0, -1  # set the result to -1
jr $ra  # return