NAME: _____ STUDENT #: _____

EECE 259: Introduction to Microcomputers **Lecture Quiz** **Feb 4, 2011**

1. Write a program that computes **(3*A-B*C)/D** and puts the answer in register r2. The symbols have been defined below. Your program should also work for other reasonable values of **A, B, C,** or **D**.

```
.equ    A, 7
.equ    D, 5
.global _start
.text
_start:









STOP:           br      STOP

.data
B:
.word   2
C:
.word   3
.end
```

2. Write a short program to compute N!, e.g. 5! = 1*2*3*4*5. Store the answer in memory at label FACT:

NIOS II Instruction Set Summary

<table>
<tr><td colspan="6">ARITHMETIC</td></tr>
<tr><td>add</td><td>addi</td><td></td><td colspan="2" rowspan="4">add <u>rC</u>, rA, rB</td><td rowspan="4">addi <u>rB</u>, rA, Imm16S

ImmS: signed</td></tr>
<tr><td>sub</td><td>subi</td><td></td></tr>
<tr><td>mul</td><td>muli</td><td></td></tr>
<tr><td>div</td><td></td><td>divu</td></tr>
<tr><td colspan="6">LOGICAL</td></tr>
<tr><td>and</td><td>andi</td><td>andhi</td><td colspan="2" rowspan="4">and <u>rC</u>, rA, rB</td><td rowspan="4">andi <u>rB</u>, rA, Imm16U
andhi <u>rB</u>, rA, Imm16U

ImmU: unsigned</td></tr>
<tr><td>or</td><td>ori</td><td>orhi</td></tr>
<tr><td>nor</td><td></td><td></td></tr>
<tr><td>xor</td><td>xori</td><td>xorhi</td></tr>
<tr><td colspan="6">SHIFT AND ROTATE</td></tr>
<tr><td>sll</td><td>slli</td><td></td><td colspan="2">shift left logical</td><td rowspan="5">sll <u>rC</u>, rA, rB
slli <u>rB</u>, rA, Imm5U
rA: value to shift
rB or Imm5U: amount of shift
(amount is always unsigned)</td></tr>
<tr><td>srl</td><td>srli</td><td></td><td colspan="2">shift right logical</td></tr>
<tr><td>sra</td><td>srai</td><td></td><td colspan="2">shift right arithmetic</td></tr>
<tr><td>rol</td><td>roli</td><td></td><td colspan="2">rotate left</td></tr>
<tr><td>ror</td><td></td><td></td><td colspan="2">rotate right</td></tr>
<tr><td colspan="6">MOVES</td></tr>
<tr><td>mov</td><td></td><td></td><td colspan="2">mov <u>rC</u>, rA</td><td>add <u>rC</u>, rA, r0</td></tr>
<tr><td>movi</td><td></td><td></td><td colspan="2">movi <u>rB</u>, Imm16S</td><td>addi <u>rB</u>, r0, Imm16S</td></tr>
<tr><td>movui</td><td></td><td></td><td colspan="2">movui <u>rB</u>, Imm16U</td><td>ori <u>rB</u>, r0, Imm16U</td></tr>
<tr><td>movia</td><td></td><td></td><td colspan="2">movia <u>rB</u>, LABEL</td><td>orhi rB, r0, high bits of LABEL
ori <u>rB</u>, rB, low bits of LABEL</td></tr>
<tr><td colspan="6">LOAD AND STORE</td></tr>
<tr><td>ldw</td><td>ldwio</td><td></td><td colspan="2">reads data memory</td><td>ldw <u>rB</u>, Imm16S(rA)</td></tr>
<tr><td>stw</td><td>stwio</td><td></td><td colspan="2">writes data memory</td><td>stw rB, <u>Imm16S(rA)</u></td></tr>
<tr><td colspan="6">JUMP AND BRANCHES</td></tr>
<tr><td>callr</td><td>call</td><td></td><td colspan="3">callr rA, call LABEL</td></tr>
<tr><td>jmp</td><td>jmpi</td><td></td><td colspan="3">jmp rA, jmpi LABEL</td></tr>
<tr><td>br</td><td>(always)</td><td></td><td colspan="3">br LABEL</td></tr>
<tr><td>beq</td><td rowspan="7">==
!=
>
>=
<
<=</td><td></td><td colspan="3" rowspan="7">beq rA, rB, LABEL</td></tr>
<tr><td>bne</td><td></td></tr>
<tr><td>bgt</td><td>bgtu</td></tr>
<tr><td>bge</td><td>bgeu</td></tr>
<tr><td>blt</td><td>bltu</td></tr>
<tr><td>ble</td><td>bleu</td></tr>
<tr><td colspan="6">COMPARISONS</td></tr>
<tr><td>cmpeq</td><td>cmpeqi</td><td></td><td></td><td rowspan="6">rA == ____
rA != ____
rA > ____
rA >= ____
rA < ____
rA <= ____
(vs. rB or Imm)</td><td rowspan="6">cmpeq <u>rC</u>, rA, rB
cmpeqi <u>rB</u>, rA, Imm16S
cmpequi <u>rB</u>, rA, Imm16U

sets <u>rC</u>=1 or <u>rB</u>=1 if comparison true, 0 otherwise</td></tr>
<tr><td>cmpne</td><td>cmpnei</td><td></td><td></td></tr>
<tr><td>cmpgt</td><td>cmpgti</td><td>cmpgtu</td><td>cmpgtui</td></tr>
<tr><td>cmpge</td><td>cmpgei</td><td>cmpgeu</td><td>cmpgeui</td></tr>
<tr><td>cmplt</td><td>cmplti</td><td>cmpltu</td><td>cmpltui</td></tr>
<tr><td>cmple</td><td>cmplei</td><td>cmpleu</td><td>cmpleui</td></tr>
</table>