Please enter your name and Athena login name in the spaces above. Enter your answers in the spaces provided after each question. You can use the extra white space and the backs of the pages for scratch work.

**Problem 1. Beta Implementation (12 points)**

Consider the assembly language program shown to the right. Assume that all register values are initialized to 0, execution starts at PC=0 and halts when HALT() is executed.

This program is run on 4 different broken Betas, where each Beta has a specified control signal stuck at the specified value, i.e., the control signal value is fixed and is not affected by the value produced by the Beta’s CTL module. For each broken Beta, please give the value in registers R1, R2, R3, and the location X: after the programs halts. **Assume that any don’t care control signal values are 0.**

### Answer Key

<table>
<thead>
<tr>
<th>Name</th>
<th>Athena login name</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elaine</td>
<td>Grant</td>
<td>Dan</td>
</tr>
<tr>
<td>WF 10, 34-301</td>
<td>WF 12, 34-303</td>
<td>WF 2, 34-303</td>
</tr>
<tr>
<td>WF 11, 34-301</td>
<td>WF 1, 34-303</td>
<td>WF 3, 34-303</td>
</tr>
<tr>
<td>Xiangyao</td>
<td>Nicole</td>
<td>Rakesh</td>
</tr>
<tr>
<td>WF 10, 13-4101</td>
<td>WF 12, 36-155</td>
<td>WF 2, 13-5101</td>
</tr>
<tr>
<td>WF 11, 13-4101</td>
<td>WF 1, 36-155</td>
<td>WF 3, 13-5101</td>
</tr>
</tbody>
</table>

### Broken control signal

<table>
<thead>
<tr>
<th>Broken control signal</th>
<th>Final value in</th>
<th>Location X:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA2SEL stuck at 0</td>
<td>42 1 0xC 0</td>
<td>0</td>
</tr>
<tr>
<td>WDSEL stuck at 0b00</td>
<td>0xF0 8 0xC 0xF0</td>
<td>0xF0</td>
</tr>
<tr>
<td>WSEL stuck at 1</td>
<td>0 0 0 -42</td>
<td></td>
</tr>
<tr>
<td>WERF stuck at 1</td>
<td>0xF0 1 0xC 42</td>
<td>42</td>
</tr>
</tbody>
</table>

Select RB instead of RC for MWD

Always write PC+4 to RC

Only write to R30 (all other regs == 0)

Write RC during ST
Problem 2. Caches (10 points)

The program from Lab 6 is shown at the right. Assume the program is being run on a Beta with a cache with the following parameters:

- 2-way set-associative
- block size of 2, i.e., 2 data words are stored in each cache line
- total number of data words in the cache is 32
- LRU replacement strategy

(A) (3 points) The cache will divide the 32-bit address supplied by the Beta into three fields: B bits of block offset (including byte offset bits), L bits of cache line index, and T bits of tag field. Based on the cache parameters given above, what are the appropriate values for B, L, and T?

value for B: __________
value for L: __________
value for T: __________

(B) (2 Points) If the MULC instruction is resident in a cache line, what will be its cache line index? the value of the tag field for the cache?

Cache line index for MULC when resident in cache: __________
Tag field for MULC when resident in cache: 0x_________

(C) (3 Points) With the values of I, A, and N as shown, list all the values j (0 ≤ j < N) where the location holding the value A[j] will map to the same cache line index as the MULC instruction in the program.

List all j where A[j] have the same cache line index as MULC: __________
A[0] @ 0x420 => cache line 4. Two words/line => A[8], A[9] are in cache line 0

(D) (2 Points) If the outer loop is run many times, give the steady-state hit ratio for the cache, i.e., assume that the number of compulsory misses as the cache is first filled are insignificant compared to the number of hits and misses during execution.

Steady-state hit ratio (%): __________
Problem 3. Pipelined Beta (8 points)

The following program fragments are being executed on the 5-stage pipelined Beta described in lecture with full bypassing, stall logic to deal with LD data hazards, and speculation for JMPs and taken branches (i.e., IF-stage instruction is replaced with a NOP if necessary). The execution pipeline diagram is shown for cycle 1000 of execution. Please fill in the diagram for cycle 1001; use “?” if you cannot tell what opcode to write into a stage. Then for both cycles use arrows to indicate any bypassing from the ALU/MEM/WB stages back to the RF stage (see example for cycle 1000 in part A).

(A) (2 points) Assume BNE is taken.

```
...  
ADDC(R1,5,R1)  
L: SUBC(R1,1,R1)  
SHRC(R0,1,R0)  
BNE(R1,L)  
ST(R1, data)  
...
```

```
Cycle | 1000 | 1001  
-------|------|------  
IF | ST | SUBC  
RF | BNE | NOP  
ALU | SHRC | BNE  
MEM | SUBC | SHRC  
WB | NOP | SUBC
```

(B) (2 points)

```
...  
ST(R31,0,BP)  
LD(BP,-12,R17)  
ADDC(SP,4,SP)  
SHLC(R17,2,R1)  
ST(R1,-4,SP)  
BEQ(R31,fact,LP)  
...
```

```
Cycle | 1000 | 1001  
-------|------|------  
IF | ST | ST  
RF | SHLC | SHLC  
ALU | ADDC | NOP  
MEM | LD | ADDC  
WB | ST | LD
```

(C) (2 points)

```
...  
XOR(R1,R2,R1)  
MULC(R2,3,R2)  
SUB(R2,R1,R3)  
AND(R3,R1,R2)  
ADD(R3,R2,R3)  
ST(R3,x)  
...
```

```
Cycle | 1000 | 1001  
-------|------|------  
IF | ADD | ST  
RF | AND | ADD  
ALU | SUB | AND  
MEM | MULC | SUB  
WB | XOR | MULC
```

(D) (2 points) Assume during cycle 1000 the DIV instruction in the RF stage triggers an ILLEGAL OPCODE (ILLOP) exception.

```
...  
LD(x,R1)  
LD(y,R2)  
SHLC(R1,3,R1)  
DIV(R2,R1,R3)  
ADDC(R3,17,R3)  
ST(R3,z)  
...
```

```
Cycle | 1000 | 1001  
-------|------|------  
IF | ADDC | ? Inst @ 0x4  
RF | DIV | NOP  
ALU | SHLC | BNE(R31,...,XP)  
MEM | NOP | SHLC  
WB | LD | NOP
```

END OF QUIZ 3!