Name_____

Rose-Hulman Institute of Technology
Electrical and Computer Engineering

EC 332 - Exam 2 Friday, April 27th, 2001

CLOSED BOOK, Open Notes. Work each problem in the space provided on its sheet. Be sure the work you present is clear so I can understand what you have done. You may use your 'C6x notes from the bookstore, the IA-64 article and a 3" by 5" card of notes. No other aids, animate or inanimate, are permitted. Please do your own work.

Problem 1 [20 points] – Your task for this problem and the next is to implement the following code on a Texas Instruments 'C6000 processor.

```
void exam2(short a[], short b[],short c[], short ans[], short n) {
   for(int i=n; i!=0; i--)
      ans[i] = (a[i] + b[i]) * c[i];
}
```

The c-code above can be implemented on the 'C6x with the following linear assembly code. The first step in optimizing the code is to draw a dependence graph for the **loop**. Draw the graph. Be sure to include everything in the graph that was discussed in class.

```
exam2:  .proc     a4,b4,a6,b6,a8
        .reserve  b3
        .reg      a_m, b_m, c_m, ans_m, a, b, c, ans, i, sum
        mv        a4, a_m
        mv        b4, b_m
        mv        a6, c_m
        mv        b6, ans_m
        mv        a8, i

loop:   ldh       *a_m++, a
        ldh       *b_m++, b
        ldh       *c_m++, c
        add       a, b, sum
        mpy       sum, c, ans
        sth       ans, *ans_m++

  [i]   sub       i,1,i
  [i]   b         loop

        .endproc  a4
```

Problem 2 – [30 points] The next step is to schedule the pipeline. Be sure to show where the prolog, loop, and epilog is. Yes I want you to show the code for all three. Revise the cycle numbers to have the correct cycle count for the code in the epilog for `n=100`.

Cycle	0	2	4	6	8	10	12	14	16
.L1									
.L2									
.M1									
.M2									
.D1									
.D2									
.S1									
.S2									
Cycle	1	3	5	7	9	11	13	15	17
.L1									
.L2									
.M1									
.M2									
.D1									
.D2									
.S1									
.S2									

Hints: You may not need all the columns. It must be scheduled as multi-cycle.

Write the code for the loop. Be sure to include all operands. Show your register and functional unit assignments. Use symbolic register names in your code (i.e. use `a_m`, not `A4`).

```
loop:
```

Variable (do only those shown)	**Register**
a_m	
b_m	
ans	
sum	
i	

Problem 3 – [25 points] Here is the 'C6000 code for the loop from problem one:

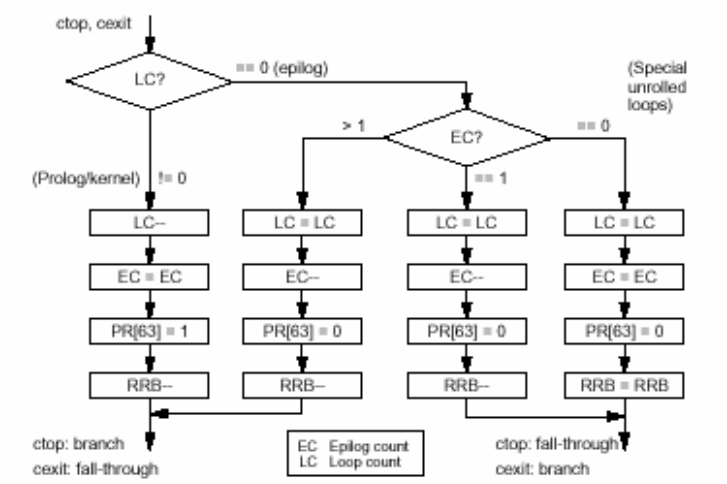
```
loop:   ldh       *a_m++, a
        ldh       *b_m++, b
        ldh       *c_m++, c
        add       a, b, sum
        mpy       sum, c, ans
        sth       ans, *ans_m++

  [i]   sub       i,1,i
  [i]   b         loop
```

a. Your task is to finish writing the IA-64 code below to do the same loop. Assume all instructions return their result on the next clock cycle. **What registers are you assuming initially point to `a[], b[]` and `c[]`?**

b. Fill in all the blanks below and answer the questions. You might want to sketch a scheduling table.

```
    mv       pr.rot   = 0       //  What is this instruction doing?

    cmp.eq   p16,p0   = r0,r0   //  To what is p0 being set?

    mov      ar.lc    = _____   //  What should the loop count be for 7 iterations?

    mov      ar.ec    = _____   //  What should the epilog count be for 7 iterations?
```

```
loop:

(____)      ld2      r____ = [r_____],2     //  Fill in all the register numbers.  Try to use a few registers as possible

(____)      ld2      r____ = [r_____],2

(____)      ld2      r____ = [r_____],2

(____)      add      r____ = r_____, r_____

(____)      mpy      r____ = r_____, r_____

(____)      st2      [r____] = r_____, 2

(____)      br.ctop.sptk.few  loop
```



Problem 4 – [25 points] Short Answer

a) **'C6000:** There are four possible *reasons* that can cause the minimum iteration interval (MII) to be greater than 1,

a. Not enough resources,

b. Live too long,

c. Loop carry path

d. Functional unit latency greater than one.

The code in problems 1-2 was multi-cycle. Which of these *reasons* caused its MII to be greater than 1? Is there more than 1 reason? Circle all the apply.

b) How is the ***SUB i,1,i*** instruction from the C6000 code in problem 1 implemented in the IA-64 code?

c) **IA-64:** The Predicate Registers on the IA-64 processor are used to implement many features of this processor. Name one feature that uses the predicate registers and *briefly* describe how the predicate registers are used.

d) **IA-64:** What is the problem with moving a regular load above a branch? How does a speculative load alleviate this problem?