Please enter your name, Athena login name, and recitation section above. Enter your answers in the spaces provided below. You can use the extra white space and the backs of the pages for scratch work.

The exam has an appendix which includes our 6.004 RISC-V reference guide.

Problem 1. Sequential Logic in BSV (15 points)

The following code implements a simple sequential circuit as a module that computes a function over a series of steps. Read the code and answer the questions about it below.

```hs
interface Diff;
  method Action start(Bit#(32) a, Bit#(32) b);
  method Bit#(32) get1;
  method Bit#(32) get2;
endinterface

module mkDiff(Diff);
  Reg#(Bit#(32)) x <- mkReg(0);
  Reg#(Bit#(32)) y <- mkReg(0);
  Reg#(Bit#(32)) i <- mkReg(0);

  rule step(y > 0);
    if (x > y-1) begin
      x <= x-y;
      i <= i+1;
    end
    else begin
      y <= 0;
    end
  endrule

  method Action start(Bit#(32) a, Bit#(32) b) if (y==0);
    i <= 0;
    x <= a;
    y <= b;
endmethod
```

1 / 20
2 / 18
3 / 20
4 / 18
5 / 24
method Bit#(32) get1 if (y == 0);
    return i;
endmethod

method Bit#(32) get2 if (y == 0);
    return x;
endmethod

endmodule

(A) (3 points) If at the beginning of step the values in the register are \( x = 21, y = 8, i = 4 \), what are the values after rule step runs once?

1. \( x = \) 
2. \( y = \) 
3. \( i = \)
(B) (6 points) If `start` is called with values `a = 14` and `b = 3`, what will the output of `get1` and `get2` be when it is ready?

1. Return value of `get1()` =
2. Return value of `get2()` =

(C) (6 points) Describe in words what function this circuit computes when the module is started with `start(a, b)`.

1. What does the return value of `get1()` correspond to when it is ready?

2. What does the return value of `get2()` correspond to when it is ready?
Problem 2. Hardware Implementation (18 points)

We want to add the following instruction to the RISC-V ISA:

\[ \text{lwr \ rd, rs1, rs2} \]

The behavior of this new `lwr` instruction is that it adds the contents of source registers `rs1` and `rs2` and uses the result as the address for your load. In other words:

\[
\text{reg[rd]} \leftarrow \text{mem[reg[rs1] + reg[rs2]]} \\
\text{pc} \leftarrow \text{pc} + 4
\]

Suppose you were given a decoder that was fully functioning prior to the addition of this new instruction, and you were told that a new type `IType LWR` has been added for this instruction. Fill in the required changes to the Decode and Execute functions in order to support adding this instruction. We have included relevant portions of these functions below to simplify this task. Assume that the rest of the hardware implementation is the same as what you have seen in lecture and lab.

(A) (12 points) Fill in the required additions to the code below.

```plaintext
typedef enum { OP, OPIMM, BRANCH, LUI, JAL, JALR, LOAD, STORE, GCD, AUIPC, LWR, Unsupported} IType deriving (Bits, Eq, FShow);

function DecodedInst decode(Bit#(32) inst);
  ...
  DecodedInst dInst = ?;
  dInst.iType = Unsupported;
  case(opcode)
    ...
    opLoad: if (funct3 == fnLW) begin
    end
    endcase
  return dInst;
endfunction

Write your code below. Please add arrows to indicate where your code fits in the decode function.
```
function ExecInst execute(DecodedInst dInst, Word rVal1, Word rVal2, 
    Word pc);

let imm = dInst.imm;
let brFunc = dInst.brFunc;
let aluFunc = dInst.aluFunc;

Word data = ?;
Word nextPc = ?;
Word addr = ?;

case (dInst.iType)
    OP: begin data = alu(rVal1, rVal2, aluFunc); nextPc = pc+4; end
    OPIMM: begin data = alu(rVal1, imm, aluFunc); nextPc = pc+4; end
    BRANCH: begin nextPc=aluBr(rVal1,rVal2,brFunc)? pc+imm : pc+4; end
    LUI: begin data = imm; nextPc = pc+4; end
    JAL: begin data = pc+4; nextPc = pc+imm; end
    JALR: begin data = pc+4; nextPc = (rVal1+imm) & ~1; end
    LOAD: begin addr = rVal1+imm; nextPc = pc+4; end
    STORE: begin data = rVal2; addr = rVal1+imm; nextPc = pc+4; end
    AUIPC: begin data = pc+imm; nextPc = pc+4; end
endcase

ExecInst eInst = ?;
eInst.iType = dInst.iType;
eInst.rd = dInst.rd;
eInst.data = data;
eInst.addr = addr;
eInst.nextPc = nextPc;
return eInst;
endfunction

Write your code below. Please add arrows to indicate where your code fits in the `execute` function.

(B) (3 points) Could a store operation that calculated the address in the same manner as `lwr` be added without major changes to the RISC-V ISA?

    Yes               No

(C) (3 points) If you answered Yes to B then provide a brief summary of the required changes. If you answered No to B then explain why not.
Problem 3. Bluespec Concurrency (9 points)

(A) (5 points) You are designing some hardware in Bluespec, and are trying to get ruleA and ruleB to execute concurrently. Both rules access register x and FIFO f:

```verilog
rule ruleA;
  f.enq(x)
endrule

rule ruleB;
  x <= f.first + 1;
  f.deq;
endrule
```

Do the following types of FIFOs allow for concurrent execution of ruleA and ruleB? Answer Yes or No.

**Pipeline FIFO:** __________

**Bypass FIFO:** __________

**Conflict-Free FIFO:** __________

Conflict Matrices for FIFOs:

<table>
<thead>
<tr>
<th>Pipeline FIFO</th>
<th>Bypass FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>enq, deq, first</td>
<td>enq, deq, first</td>
</tr>
<tr>
<td>enq</td>
<td>C</td>
</tr>
<tr>
<td>deq</td>
<td>&lt;</td>
</tr>
<tr>
<td>first</td>
<td>&lt;</td>
</tr>
</tbody>
</table>

Conflict-Free FIFO

<table>
<thead>
<tr>
<th>enq, deq, first</th>
</tr>
</thead>
<tbody>
<tr>
<td>enq</td>
</tr>
<tr>
<td>deq</td>
</tr>
<tr>
<td>first</td>
</tr>
</tbody>
</table>
Problem 4. Caches (24 points)

Consider the diagram to the right for a 2-way set associative cache to be used with our RISC-V processor. Each cache line holds a single 32-bit word of data along with its associated tag and valid bit (0 when the cache line is invalid, 1 when the cache line is valid).

(A) (2 points) RISC-V uses 32-bit byte addresses, A[31:0]. To ensure the best cache performance, which address bits should be used for the cache index? Which address bits should be used for the tag field?

address bits used for cache index: A[_____:_____]  
address bits used for tag field: A[_____:_____]  

(B) (2 points) Suppose the processor does a read of location 0x5678. Identify which cache location(s) would be checked to see if that location is in the cache. For each location, specify the cache way (A or B) and set (row) number (0 through 7). E.g., 3A for set 3, way A. If there is a cache hit on this access, what would be the contents of the tag data for the cache line that holds the data for this location?

cache location(s) checked on access to 0x5678: ________________________  
cache tag data on hit for location 0x5678 (hex): 0x_____________

(C) (2 points) Assume that checking the cache on each read takes 1 cycle and that refilling the cache line on a miss takes an additional 8 cycles. If we wanted the average access time over many reads to be 1.1 cycles, what is the minimum hit ratio the cache must achieve during that period of time? You needn’t simplify your answer.

minimum hit ratio for 1.1 cycle average access time:______________
(D) (5 points) Find the cache hit ratio for the following program. Because the loop is executed many times, you can ignore code before/after the loop and focus on the average behavior of a loop iteration. Assume the cache is empty before execution begins (all valid bits are 0) and that an LRU replacement policy is used. Remember the cache is used for both instruction and data accesses.

```
. = 0  // Code starts at address 0
addi x4, x0, 0x100
mv x1, x0
lui x2, 1  // x2 = 0x1000
loop: lw x3, 0(x4)
    addi x4, x4, 4
    add x1, x1, x3
    addi x2, x2, -1
    bnez x2, loop
    sw x1, 0x100(x0)
    unimp  // halt
.
  = 0x100
source:
. = . + 0x4000  // Set source to 0x100, reserve 0x1000 words
```

hit ratio: ______________

(E) (3 points) After the program of part (D) has finished execution what information is stored in set (row) 4 of the cache? Give the addresses for the two locations that are cached (one in each of the sections) or briefly explain why that information can’t be determined.

Addresses whose data is cached in set 4: ___________ and ___________

For the rest of this problem, the cache is modified so that it has a block size of 2 words per line and the number of words in the cache remains the same (i.e., the cache now has 4 lines per way).

(F) (2 points) Specify which bits of the address should be used for the block offset, the cache index and the tag.

address bits used for block and byte offset: A[____:____]

address bits used for cache index: A[____:____]

address bits used for tag field: A[____:____]

(G) (4 points) Find the hit ratio for the program in part (D) using this new cache configuration.

hit ratio: ______________

END OF PRACTICE QUIZ 2!