Problem #1: For this problem, assume a VLIW processor with three integer units (X, Y, Z), one multiply unit (M), and two load-store units (LS0, LS1). ALU instructions have a latency of 1, multiply instructions have a latency of 5, and loads have a latency of 3. One branch can execute per cycle and executes in the Z pipeline. For the following code, fully unroll and software pipeline the code to improve performance. Show the prolog and epilog code. Feel free to use more registers, reorder code, and rename registers for performance. What high order functionality does the function implement? What is the average rate of multiplies per cycle in the middle of the loop?

//R0 is hardwired to zero
//R31 is the link register
//R1 contains pointer to input array (elements are word sized)
//R2 contains pointer to output array (elements are word sized)
// (Output array is larger than input array)
//R3 contains the size of the input array in words
function:
ADDI R16, R0, 0x456
ADDI R17, R0, 0x789
ADDI R18, R0, 0x901
ADD R4, R0, R0
ADD R5, R0, R0
loop:
LW R6, 0(R1)
MUL R8, R4, R16
MUL R9, R5, R17
MUL R10, R6, R18
ADDI R1, R1, 4
SUBI R3, R3, 1
ADD R8, R8, R9
ADD R8, R8, R10
ADD R4, R5, 0
ADD R5, R6, 0
SW R8, 0(R2)
ADDI R2, R2, 4
BNEZ R3, loop
MUL R8, R4, R16
MUL R9, R5, R17
ADD R8, R8, R9
SW R8, 0(R2)
ADDI R2, R2, 4
ADD R4, R5, 0
MUL R8, R4, R16
SW R8, 0(R2)
JR R31
Problem #2: For this problem, assume a VLIW processor with three integer units (X, Y, Z), one multiply unit (M), and two load-store units (LS0, LS1). ALU instructions have a latency of 1, multiply instructions have a latency of 5, and loads have a latency of 2. One branch can execute per cycle and executes in the Z pipeline. The following code has been bundled assuming the EQ scheduling model. What is the value of R12, R13, and R14 after this code executes? Not changing register names, reschedule this code assuming the LEQ model. Why is the LEQ model more flexible?

{ADDI R9, R0, 9; ADDI R10, R0, 10;}
{ADDI R6, R0, 6; ADDI R8, R0, 8; ADDI R5, R0, 5;}
{LW R6, 0(R7); LW R8, 4(R7);}
{ADDI R12, R6, 1; ADDI R13, R8, 2;}
{MUL R7, R6, R9;}
{MUL R5, R8, R10;}
{LW R14, 8(R7);} // Assume that 8(R7) contains the value 0x1
{ADD R15, R16, R17;}
{ADD R14, R14, R5;}
{SUB R19, R18, R22;}
{ADD R5, R7, R5;}

Problem #3: Rewrite the following code assuming the instruction set has been augmented with conditional move instructions, movz and movn. Assume that the branch misprediction penalty is 10 cycles and that the branch to “forward” is random and data-dependent. Does it make sense to predicate? Movz and movn have the following semantics:

movz rd, rs, rt if ( R[rt] == 0 ) then R[rd] <- R[rs]
movn rd, rs, rt if ( R[rt] != 0 ) then R[rd] <- R[rs]

Code Sequence for problem 3:

ADDI R6, R0, 1
ADDI R3, R0, 50
loop:
LW R8, 0(R9)
BEQZ R8, forward
ADD R12, R15, R8
SUB R24, R24, R12
J done
forward:
ADDI R24, R24, 10
done:
SUBI R3, R3, 1
BNEZ R3, loop
Problem #4: Assume a computer architecture with a 1024-entry Branch History Table (BHT). Each BHT entry is a two bit predictor that implements a two-bit saturating counter that starts in the strong not-taken state. After the following code execution, what is the state of the BHT? What is the prediction accuracy of each branch?

p_4:
// Assume that at address 0x1000, there is an array of word sized integers
// with the following data [0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1, 0x0, 0x1]
ADDI R7, R0, 7
ADDI R11, 0x1000
loop:
SUBI R7, R7, 1
ANDI R8, R7, 1
b_0:
BEQZ R8, l_1
LW R12, 0(R11)
b_1:
BEQZ R12, l_2
l_1:
ADD R9, R9, R16
l_2:
ADDI R11, R11, 4
b_3:
BNEZ R7, loop

Problem #5: Assume a computer architecture has a 11-bit global Branch History Register (BHR) that is used to index into a 2048-entry Pattern History Table (PHT), where each PHT entry is a single bit predictor. Assume that function p_4 in the code for problem 4 (above) is executed 50 times. On the 51st execution, what is the branch prediction accuracy for the branch b_1? Draw the final state of the global BHR.

Problem #6: Branch Target Buffers (BTBs) are used to determine the address of the target of branch or jump. At aggressive clock frequencies why are BTBs used? When is the destination of a branch typically known in a 5 stage pipeline with a dedicated branch address adder? Does a branch address adder help for JALR? Does a BTB help with determining the target of a JALR?

Problem #7: Page 136 in H&P5, Problem 2.8 parts a, b, and c.

Problem #8: Page 138 in H&P5, Problem 2.11 parts a and b.

Problem #9: What is the reach of a 16 entry fully associative TLB assuming that there are two valid page sizes, 4KB and 1MB?
Problem #10: You are designing the page tables for a processor with a 64-bit virtual address space. The top bit is reserved and is always set to be zero, therefore there is effective 63-bit virtual address space. The processor has a 64-bit physical address space. Assuming a page size of 8KB, construct a multi-level page table where the different levels of the page table naturally fit within an 8KB page. Assume that each leaf page table entry needs a valid bit, a dirty bit, a kernel/supervisor bit, and two software reserved bits. Assuming that the OS dedicates 10 pages to page table entries (any level) to a particular process, what is the maximum amount of physical memory that can be addressed by that process? What is the minimum?

Problem #11: On a machine with a software-managed memory management unit (MMU) when a TLB miss occurs, what are the possible reasons? Does this always result in a bus-error/segmentation fault? On a machine with a hardware managed MMU with hardware page-table walker, does a page fault always result in a bus-error/segmentation fault?

Problem #12.1: You are executing on a VMIPS (as described on p. 266 of H&P5) architecture the code below. Assume that the maximum vector length of the architecture is 128. Draw the pipeline diagram of this code executing on a single-lane architecture which has an independent load unit, store unit, multiply unit, and ALU unit. Loads have a latency of three cycles (L0, L1, L2), stores take two cycles to occur (S0, S1), multiplies take 5 cycles (Y0, Y1, Y2, Y3, Y4), and ALU operations take two cycles (X0, X1). Assume full pipelining of the functional units. Assume that the pipeline has a dedicated register read stage and a single write-stage. For the first part of this problem, assume that the architecture supports chaining through the register file, but only has two read ports and one write port on the register file.

Problem #12.2: Redo the above pipeline diagram assuming that the pipeline has a write port and two read ports per functional unit and that the architecture has two lanes (two duplicates of all functional unit resources).

C Code:

```c
for (i = 0; i < 6; i++)
{
    a[i] = (b[i] * 7) + c[i];
}
```

VMIPS:

```c
ADDI R7, R0, 7
CVT.W.D F2, R7
ADDI R1, R0, 6
MTC1 VLR, R1
LV V1, R4
MULVS.D V3, V1, F2
LV V2, R5
ADD V4, V3, V2
SV R6, V4
```

Problem #13: Do GPUs have vector length registers? Describe how GPUs handle the case where two elements in a vector of data need different processing.

Problem #14: Page 254 in H&P5, Problem 3.13
Problem #15: Assume that your architecture has a test-and-set instruction as its only atomic primitive. Implement atomic compare-and-exchange out of the test-and-set primitive.

Problem #16: List the possible sequentially consistent outcomes for the variables i and j after the completion of executing the three threads T1, T2, and T3. Assume that all threads begin executing after ‘i’ has been set to 9 and ‘j’ is set to 10.

T1:
ADDI R1, R0, 30
SW R1, 0(i)
LW R2, 0(j)
SW R2, 0(j)

T2:
ADDI R5, R0, 99
LW R6, 0(j)
ADD R7, R5, R6
SW R7, 0(j)

T3:
ADD R8, R0, 100
SW R8, 0(i)

Problem #17: You are writing a multi-threaded program that will count the number of occurrences of a value in an array. The values in the array are between 0 and 1023. In effect, you will be building a histogram. Assume that the list of numbers is very large, on the order of gigabytes large. Extend the following program such that 100 threads (processors) can execute on the program concurrently. Assume a sequentially consistent memory model. Add P() and V() semaphores where appropriate and add any storage needed for the semaphores. Explain why the speedup of such a solution may not be 100x.

// Sequential code, assume that the input and output arrays are created // outside of the function
#define MAX_VALUE 1023
function(int input_array_size, int * input_array, int * output_array)
{
    int counter;
    for (counter = 0; counter < input_array_size; counter++)
    {
        assert(input_array[counter] <= MAX_VALUE);
        assert(input_array[counter] >= 0);
        output_array[input_array[counter]]++;
    }
}
Problem #18: Show for each cache line and cache what state it is in on every cycle assuming three processors executing code as interleaved below. Assume a 64-byte cache line block size. Assume all cores contain a direct mapped cache that is 4KB large. First, assume that the processors are using a snoopy MSI cache coherence protocol. Second, repeat this for a MESI protocol.

<table>
<thead>
<tr>
<th>Time</th>
<th>P1:</th>
<th>P2:</th>
<th>P3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LW R1, 4(R0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>LW R1, 16(R0)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>LW R1, 4(R0)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>SW R2, 100(R0)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>LW R4, 104(R0)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>LW R3, 100(R0)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SW R1, 0(R0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>LW R1, 4100(R0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>SW R2, 4100(R0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>SW R3, 4100(R0)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>SW R5, 0(R0)</td>
</tr>
</tbody>
</table>

Problem #19: Calculate the bisection bandwidth for a 4-ary 3-cube without end-around, but where each link is 32-bits wide and clocks at 800MHz. Calculate the bisection bandwidth of an 8-node omega network with 64-bit links that clock at 1.2GHz.

Problem #20: How large of a credit counter is needed to provide full bandwidth on a link where the link has one cycle for routing delay, two cycles for link delay, and the return credit takes two cycles? What is the bandwidth as a proportion of the maximum if the credit size is two smaller than the needed number?

Problem #21: Assume that a message is routed on a 2D dimension-ordered network that is 4 by 4. Assume that the link delay is one cycle and that the router delay in each hop is two cycles. Assume that each link is one byte wide. Assume that the flit length is 4 bytes and the phit size is one byte. How many cycles does it take to send a 32-byte message from location (0,0) to location (2,3) assuming no insertion or destination delay assuming that the architecture implements store-and-forward? Repeat assuming that the network is a wormhole/cut-through switched network.
Problem #22: Show for each cache line, cache, and directory controller what state it is on every load/store. Assume that the code is executing on three processors as interleaved below. Assume that there is one centralized directory. Also, draw the share list that exists in the directory. Assume a 64-byte cache line block size. Assume all cores contain a direct mapped cache that is 4KB large. Assume that a MSI protocol is used in the caches and a ESU protocol is used at the directory.

<table>
<thead>
<tr>
<th>Time</th>
<th>P1:</th>
<th>P2:</th>
<th>P3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LW R1, 4(R0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>LW R1, 16(R0)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>LW R1, 4(R0)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>SW R2, 100(R0)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>LW R4, 104(R0)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>LW R3, 100(R0)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SW R1, 0(R0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>LW R1, 4100(R0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>SW R2, 4100(R0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>SW R3, 4100(R0)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>SW R5, 0(R0)</td>
</tr>
</tbody>
</table>

Problem #23: Page 420 in H&P5, Problem 5.11