1. Write a MIPS program fragment for the following high-level language statement:
   \[ a = b + 100; \]
   Assume that \( a \) is in register \( $t0 \) and \( b \) is in register \( $t1 \).

2. Write a MIPS program fragment for the following high-level language statement:
   \[ g = h + A[14]; \] // Assume \( A \) is an int array with each element taking 4 bytes
   Assume that \( g \) is in register \( $t0 \), \( h \) is in register \( $t1 \), and the base address of \( A \) is in register \( $s1 \).

3. Write a MIPS program fragment for the following high-level language statement:
   \[ A[275] = g + A[275]; \] // Assume \( A \) is an int array with each element taking 4 bytes
   Assume that \( g \) is in register \( $t0 \), and the base address of \( A \) is in register \( $s1 \).

4. Write a MIPS program fragment for the following high-level language statement:
   \[ g = h + A[i]; \] // Assume \( A \) is an int array with each element taking 4 bytes
   Assume that \( g \) is in register \( $t0 \), \( h \) is in register \( $t1 \), the base address of \( A \) is in register \( $s1 \), and \( i \) is in register \( $s2 \).

5. Write a MIPS program fragment for the following high-level language statement:
   \[ x[10] = x[11] + c; \] // Assume \( x \) is an int array with each element taking 4 bytes
   Assume that \( c \) is in register \( $t0 \), and the base address of \( x \) is 4,000,000 in base 10. (Hint: You should use the Load Upper Immediate or \( lui \) instruction).

6. (a) Write a MIPS program fragment for the following high-level language statement:
   \[ B[350] = g + B[350]; \] // Assume \( B \) is an int array with each element taking 4 bytes
   Assume that \( g \) is in register \( $s1 \) and that the base address of the array \( B \) is in register \( $s2 \).

   Assume that registers \( $s0 \) through \( $s7 \) map to register numbers 16 through 23, and registers \( $t0 \) through \( $t7 \) map to register numbers 8 through 15. Use the instruction format information provided below to convert your instructions in Part (a) to machine language code. First express
your machine language code using decimal numbers for the fields and then convert into Binary and eventually into hexadecimal representation for each instruction.

(b) Instructions for the program in Part (a) converted into Decimal

(c) Conversion of Part (b) into Binary

(d) Conversion of Part (c) into Hexadecimal

Instruction Format of relevant MIPS instructions

<table>
<thead>
<tr>
<th>6 bits</th>
<th>5 bits</th>
<th>5 bits</th>
<th>5 bits</th>
<th>5 bits</th>
<th>6 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>rd, rs, rt</td>
<td>0</td>
<td>rs</td>
<td>rt</td>
<td>rd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6 bits</th>
<th>5 bits</th>
<th>5 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>lw</td>
<td>rt, d (rs)</td>
<td>35</td>
<td>rs</td>
</tr>
<tr>
<td>sw</td>
<td>rt, d (rs)</td>
<td>43</td>
<td>rs</td>
</tr>
</tbody>
</table>

7. Add comments to the following MIPS code and describe in one sentence what it computes.
Assume that $a0$ is used for the input and initially contains $n$, a positive integer. Assume that $v0$ is used for the output.

begin: addi $t0, $zero, 0
addi $t1, $zero, 1
loop: slt $t2, $a0, $t1
bne $t2, $zero, finish
add $t0, $t0, $t1
addi $t1, $t1, 2
j loop
finish: add $v0, $t0, $zero

8. Write MIPS code for the following high-level language program fragment:

Loop:  g = g + A[i]; // Assume g is in $s1; i is in $s3; base address of A is in $s5
i = i + j;    // Assume j is in register $s4; h is in register $s2
if (i != h) go to Loop
9. Describe the contents of registers $t0$ and $t2$ when the following MIPS program reaches the label marked “done”.

Register $s0$ contains the base address of an int array B of size 100. Assume that integer values have been inputted into the array already.

```
lw     $t0,  0($s0)
addi   $t1,  $s0,  40
addi   $s0,  $s0,  4
loop:  beq   $s0,  $t1, done
lw     $t2,  0($s0)
blt    $t2,  $t0,  skip    # If $t2 is less than $t0, go to skip
addi   $t0,  $t2,  0
skip:  addi   $s0,  $s0,  4
j      loop
done:  . . . .
```

10. The following program tries to copy words from the address in register $a0$ to the address in register $a1$, counting the number of words copied in register $v0$. The program stops copying when it finds a word equal to 0. You do not have to preserve the contents of registers $v1$, $a0$, and $a1$. This terminating word should be copied but not counted.

```
Loop: lw     $v1,  0($a0)  # Read next word from source
       addi   $v0,  $v0,  1  # Increment count of words copied
       sw     $v1,  0($a1)  # Write to destination
       addi   $a0,  $a0,  1  # Advance pointer to next source
       addi   $a1,  $a1,  1  # Advance pointer to next destination
       bne    $v1,  $zero, Loop # Loop if word copied is ≠ 0
```

There are multiple bugs in this MIPS program. You should fix all the bugs and turn this program into a bug-free version.

11. Consider the following fragment of high-level language code:

```
for (i = 0; i <= 100; i = i + 1)
{
    a [i] = b[i] + c;
} // end for i
```

Assume that a and b are integer arrays (each element is stored in 4 bytes) and the base address of a is in $a0$ and the base address of b is in $a1$. The loop variable i is in register $t0$ and the constant c is in register $s0$.

(a) Write MIPS code for this high-level language program fragment.
(b) How many MIPS instructions are executed during the running of your code?
(c) How many memory data references will be made during execution of your code?
Consider the following **while** loop in a high-level language:

```java
while ( k == a[i] ) // k is in register $s5
    { // i is in register $s3
        i = i + j; // j is in register $s4
            // base address of integer array a is in register $s6
    } // end while
```

The MIPS assembly language program given below implements the above loop.

```
Loop: add $t1, $s3, $s3  # t1 ← i + i
    add $t1, $t1, $t1  # t1 ← 4*i
    add $t1, $t1, $s6  # t1 ← address of a[i]
    lw $t0, 0 ($t1)  # t0 ← a[i]
    bne $t0, $s5, Exit
    add $s3, $s3, $s4  # i ← i + j
j Loop
```

Exit: . . .

Assume that in the while loop, a[i + m*j] is equal to k for values of 0 <= m <= 9 and is not equal to k when m = 10. Therefore, 10 iterations of the loop are executed.

(a) How many assembly language instructions are executed in the MIPS program given above for 10 iterations of the while loop?

(b) Write a semantically equivalent MIPS assembly language program to reduce by more than half the total number of instructions executed for 10 iterations of the while loop.

### 13. Convert each of the following into a 32-bit two’s complement binary number:

(a) 512<sub>ten</sub>
(b) −1,023<sub>ten</sub>
(c) −4,000,000<sub>ten</sub>

### 14. Write the decimal number represented by each of the following two’s complement binary numbers:

(a) (1111 1111 1111 1111 1110 0000 1100)<sub>two</sub>
(b) (1111 1111 1111 1111 1111 1111 1111)<sub>two</sub>
(c) (0111 1111 1111 1111 1111 1111 1111)<sub>two</sub>

### 15. For the hexadecimal number 0x 7FFF FFFA, what binary number does it represent?

### 16. Write the hexadecimal number represented by the following binary number:

(1100 1010 1111 1110 1111 1010 1100 1110)<sub>two</sub>
17. Two friends, Harry and David, are arguing. Harry says, “All integers greater than zero and exactly divisible by six have exactly two 1s in their binary representation.” David disagrees. He says, “No, but all such numbers have an even number of 1s in their binary representation.” Do you agree with Harry or with David, or with neither? Justify your answer with a proof or counterexamples.

18. Bits in the memory of a computer have no inherent meaning. They mean something only under an interpretation.

Given the bit pattern:

1000 1111 1111 1100 0000 0000 0000

what does it represent under each of the following interpretations:

(a) a two’s complement integer?
(b) an unsigned integer?
(c) a MIPS instruction?

19. For the MIPS single cycle Datapath, complete the following Control Table with entries 0, 1, or D for Don’t care. Be very careful when using a Don’t care control signal to avoid unintended consequences. The instructions appear in the column headings and the control variables are in the rows of the table. Assume that the BEQ instruction is a branch that is taken. The last row for the ALUOp has been completed for you with the name of the operation done in the ALU.

<table>
<thead>
<tr>
<th>Add</th>
<th>Sub</th>
<th>LW</th>
<th>SW</th>
<th>BEQ</th>
<th>Addi</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Taken)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RegDst
ALUSrc
MemToReg
RegWrite
PCSrc
MemWrite
MemRead
ALUOp

Add | Sub | Add | Add | Sub | Add
20. The format of a proposed \texttt{NewLW} instruction in an R-type-like format is as follows:

\[
\text{NewLW} \quad \text{rd, rs (rt)} \quad \text{Opcode} \quad \text{rs} \quad \text{rt} \quad \text{rd} \quad \ldots \quad \# \text{R[rd]} \leftarrow \text{MEM} [\text{R[rs]} + \text{R[rt]}] \\
\text{6 bits} \quad \text{5 bits} \quad \text{5 bits} \quad \text{5 bits}
\]

What changes, if any, would you make to the single cycle datapath to execute the \texttt{NewLW} instruction?

Indicate the necessary changes and also specify the values of the multiplexor variables RegDst, AluSrc, and MemToReg, and the values of the RegWrite, MemRead, and MemWrite control variables for this instruction.

21. For the MIPS multi-cycle datapath, draw the finite state machine that represents the control necessary for the execution of the instructions \texttt{NewLW} described in problem 20, and the instruction \texttt{WhereAmI} (WAI) which does the following: It puts the instruction’s location, that is, the value of the PC when the instruction was called into a register specified by the rt field (i.e., bits [20…16]).

You should number all the states, label the states with actions taken from the set \{IF, ID, EX, MEM, WB\} and also indicate in each state the values of all the relevant multiplexor variables and control signals. Complete the diagram that has been started below.