Assignment 2 Solutions

Instruction Set Architecture, Performance and Other ISAs

Due: April 17, 2014

Unless otherwise noted, the following problems are from the Patterson & Hennessy textbook (5th ed.).

1. Translation between C and MIPS:

   (1) C to MIPS. Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.
   C Code:
   f = g + A[B[4]-B[3]];

   - For the C statement above, what is the corresponding MIPS assembly code?

   Ans:
   ```
   lw $t0, 16($s7)     // $t0 = B[4]
   lw $t1, 12($s7)     // $t1 = B[3]
   sub $t0, $t0, $t1  // $t0 = B[4] – B[3]
   sll $t0, $t0, 2        //  $t0 = $t0 * 4
   add $t0, $t0, $s6   //  $t0 = &A[B[4] – B[3]]
   lw $t1, 0($t0)        // $t1 = A[B[4] – B[3]]
   add $s0, $s1, $t1   // f = g + A[B[4] – B[3]
   ```

   (2) MIPS to C. Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.
   MIPS Code:
   ```
   sll $t0, $s0, 2
   add $t0, $s6, $t0
   sll $t1, $s1, 2
   add $t1, $s7, $t1
   lw $s0, 0($t0)
   addi $t2, $t0, 4
   lw $t0, 0($t2)
   add $t0, $t0, $s0
   ```

```
sw $t0, 0($t1)
```

- For the MIPS assembly instructions above, what is the corresponding C statement?

Ans:

```
sll $t0, $s0, 2         // $t0 = f * 4;
add $t0, $s6, $t0       // $t0 = &A[f];
sll $t1, $s1, 2         // $t1 = g * 4;
add $t1, $s7, $t1       // $t1 = &B[g];
lw $s0, 0($t0)          // f = A[f];
addi $t2, $t0, 4        // $t2 = &A[f + 1];        // Note here f is still of the original value.
lw $t0, 0($t2)          // $t0 = A[f + 1];
add $t0, $t0, $s0       // $t0 = A[f + 1] + A[f];  // Here assumes a nop.
sw $t0, 0($t1)          // B[g] = A[f + 1] + A[f];
```

Overall:
B[g] = A[f + 1] + A[f];
f = A[f];
The two statements cannot swap order.

2. Pseudo-instructions: Show how to implement the following pseudo-instructions as they would be implemented in a real machine (e.g., using $at):

    1. nop # Do nothing (show three different ways)
    2. li $s0, <32-bit constant> # Load 32-bit immediate value into s0
    3. div $s0, $s1, $s2 # Integer division: s0 = s1/s2

Ans:

1. There are many possible ways. For example:
```
add $zero, $zero, $zero
sll  $zero, $zero, $zero
or $zero, $zero, $zero
```

2. This instruction can be implemented a couple of different ways:
```
lui $s0, <16 upper bits>
ori $s0, $s0, <16 lower bits>
```
or:
```
lui $s0, <16 upper bits>
addi $s0, $s0, <16 lower bits>
```

3. New instructions: Implement register indirect conditional branches (beqr and bner) as pseudo-instructions. Give a proposal for adding them to the ISA (i.e., describe how they could be encoded in the I-Type, R-Type, or J-Type format, and propose an opcode for them (use Figure A.10.2 MIPS opcode map)). Give a (brief) argument for or against their inclusion.

   Ans:
   One way to implement beqr $s0, $s1, $s2:

```
        bne $s0, $s1, skip     // if (s0 != s1) goto skip
        nop                    // nop
        jr $s2                 // branch to $s2
        nop                    // nop
skip:
```

   bner is the same, just replace bne with beq.

   This instruction can be encoded as a R-Type instruction, because it has three register parameters. We can use opcode 0, and any funct value that has not yet been assigned.

   Arguments for:
   - Reduces code size
   - Implements in HW could save a branch

   Arguments against:
   - Increase processor complexity
   - Might slow down decode login or other parts of the pipeline
   - Can be implements with existing instructions

4. Performance:
   Consider three different processors, P1 P2 and P3, executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and a CPI of 2.2.

   - 1. Which processor has the highest performance expressed in instructions per second?
   - 2. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.
   - 3. We are trying to reduce the execution time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

   Ans:
   1. P1: 3GHz / 1.5 = 2 * 10^9 instructions per second
      P2: 2.5GHz / 1.0 = 2.5 * 10^9 instructions per second

P3: 4GHz / 2.2 = 1.82 * 10^9 instructions per second
So P2 has the highest performance among the three.

2. Cycles:
   P1: 3GHz * 10 = 3 * 10^10 cycles
   P2: 2.5GHz * 10 = 2.5 * 10^10 cycles
   P3: 4GHz * 10 = 4 * 10^10 cycles

   Num of instructions:
   P1: 3GHz * 10 / 1.5 = 2 * 10^10 instructions
   P2: 2.5GHz * 10 / 1.0 = 2.5 * 10^10 instructions
   P3: 4GHz * 10 / 2.2 = 1.82 * 10^10 instructions

3. Execution time = (Num of instructions * CPI) / (Clock rate)
   So if we want to reduce the execution time by 30%, and CPI increases by 20%, we have:
   Execution time * 0.7 = (Num of instructions * CPI * 1.2) / (New Clock rate)
   New Clock rate = Clock rate * 1.2 / 0.7 = 1.71 * Clock rate
   New Clock rate for each processor:
   P1: 3GHz * 1.71 = 5.13 GHz
   P2: 2.5GHz * 1.71 = 4.27 GHz
   P3: 4GHz * 1.71 = 6.84 GHz