# Homework #1

**Assigned**: 19/03/2020

**Due**: 27/03/2020

1. (**20 pts**) Consider the following table that describes an implementation of an instruction set architecture and the occurrence frequency of the instructions in a given benchmark.

|  |  |  |
| --- | --- | --- |
| **Instruction Class** | **clock cycles (CPI)** | **Frequency** |
| A | 5 | 25% |
| B | 6 | 10% |
| C | 2 | 20% |
| D | 8 | 10% |
| E | 4 | 35% |

* 1. Compute the overall CPI for the given benchmark. (**5 pts**)

CPI = 5×0.25 + 6×0.1 + 2×0.20 + 8×0.1 + 4×0.35 = 4.45

* 1. The clock frequency is 3.2 GHz and the number of instructions in the benchmark is 15 billion. Compute the execution time of the benchmark. (**5 pts**)

T = s

* 1. Assume that you are given an opportunity of reducing the CPI of any one class of instruction to 1. You can select at most one instruction to improve. Which instruction class would you choose and how much can the performance be improved? (**10 pts**)

If we improve class A: CPI = 1×0.25 + 6×0.1 + 2×0.20 + 8×0.1 + 4×0.35 = 3.45

If we improve class B: CPI = 5×0.25 + 1×0.1 + 2×0.20 + 8×0.1 + 4×0.35 = 3.95

If we improve class C: CPI = 5×0.25 + 6×0.1 + 1×0.20 + 8×0.1 + 4×0.35 = 4.25

If we improve class D: CPI = 5×0.25 + 6×0.1 + 2×0.20 + 1×0.1 + 4×0.35 = 3,75

If we improve class **E**: CPI = 5×0.25 + 6×0.1 + 2×0.20 + 8×0.1 + 1×0.35 = **3.40**

1. (**30 pts**) Consider Booth’s algorithm below for multiplying integers including signed ones (two’ complement).

**00 or 11**

**Start**

**i := 0**

**a-1 := 0**

**C := 0**

**aiai-1**

**CH := CH - B**

**CH := CH + B**

**10**

**01**

**C := C >> 1   
i := i + 1**

**i=n**

**stop**

**1**

**0**

C = A × B, C is 2n-bit, A and B are n-bit registers. CH is upper n-bit of C register.

Using Booth’s algorithm with n = 4, do the following multiplication operations:

6 × 5 = ?

-5 × -4 = ?

-4 × 7 = ? (**10 pts each**)

Shows the steps of the algorithm.

6 × 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **i** | **B** | **A** | **aiai-1** | **operation** | **C** |
| **0** | **0101** | **0110** | **00** | **CH := CH** | **0000 0000** |
| **C := C >> 1** | **0000 0000** |
| **1** | **0101** | **0110** | **10** | **CH := CH - B** | **1011 0000** |
| **C := C >> 1** | **1101 1000** |
| **2** | **0101** | **0110** | **11** | **CH := CH** | **1101 1000** |
| **C := C >> 1** | **1110 1100** |
| **3** | **0101** | **0110** | **01** | **CH := CH + B** | **0011 1100** |
| **C := C >> 1** | **0001 1110** |

(-5) × (-4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **i** | **B** | **A** | **aiai-1** | **operation** | **C** |
| **0** | **1100** | **1011** | **10** | **CH := CH - B** | **0100 0000** |
| **C := C >> 1** | **0010 0000** |
| **1** | **1100** | **1011** | **11** | **CH := CH** | **0010 0000** |
| **C := C >> 1** | **0001 0000** |
| **2** | **1100** | **1011** | **01** | **CH := CH + B** | **1101 0000** |
| **C := C >> 1** | **1110 1000** |
| **3** | **1100** | **1011** | **10** | **CH := CH - B** | **0010 1000** |
| **C := C >> 1** | **0001 0100** |

(-4) × (7)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **i** | **B** | **A** | **aiai-1** | **operation** | **C** |
| **0** | **0111** | **1100** | **00** | **CH := CH** | **0000 0000** |
| **C := C >> 1** | **0000 0000** |
| **1** | **0111** | **1100** | **00** | **CH := CH** | **0000 0000** |
| **C := C >> 1** | **0000 0000** |
| **2** | **0111** | **1100** | **10** | **CH := CH - B** | **1001 0000** |
| **C := C >> 1** | **1100 1000** |
| **3** | **0111** | **1100** | **11** | **CH := CH** | **1100 1000** |
| **C := C >> 1** | **1110 0100** |

1. (**30 pts**) Consider the following C language statements.

* f = -g + h + B[i]+ C[j] (**10 pts**)
* f = A[B[g]+ C[h] + j] (**20 pts**)

Assume that the local variables f, g, h, i and j of integer types (32-bit) are assigned to registers $s0, $s1, $s2, $s3 and $s4 respectively. Assume also the base address of the arrays A, B and C of integer types are in registers $s5, $s6 and $s7, respectively (i.e. $s0 🡪 f, $s1 🡪 g, $s2 🡪 h, $s3 🡪 i, $s4 🡪 j, $s5 🡪 &A[0] , $s6 🡪 &B[0], $s7 🡪 &C[0]).

For the C statements above, provide MIPS Assembly instructions.

* **f = -g + h + B[i]**

**sll $t0, $s3, 2 #$t0 = 4i**

**add $t0, $t0, $s6 #$t0 = &B[i]**

**lw $s0, 0($t0) #$s0 = B[i]**

**sll $t1, $s4, 2 #$t1 = 4j**

**add $t1, $t1, $s6 #$t1 = &C[j]**

**lw $t1, 0($t1) #$s0 = C[j]**

**add $s0, $s0, $t1 #$s0 = B[i] + C[j]**

**sub $s0, $s0, $s1 #$s0 = B[i] + C[j] - g**

**add $s0, $s0, $s2 #$s0 = B[i] + C[j] – g + h**

* **f = A[B[g]+ C[h] + j]**

**sll $t0, $s1, 2 #$t0 = 4\*g**

**add $t0, $t0, $s6 #$t0 = &B[g]**

**lw $t0, 0($t0) #$t0 = B[g]**

**sll $t1, $s2, 2 #$t1 = 4\*h**

**add $t1, $t1, $s7 #$t1 = &C[h]**

**lw $t1, 0($t1) #$t1 = C[h]**

**add $t0, $t0, $t1 #$t0 = B[g] + C[h]**

**add $t0, $t0, $s4 #$t0 = B[g] + C[h] + j**

**sll $t0, $t0, 2 #$t0 = 4×(B[g] + C[h] + i)**

**add $t0, $t0, $s5 #$t0 = &A[B[g] + C[h] + i]**

**lw $s0, 0($t0) #$s0 = A[B[g]+ C[h] + j]**

1. (**20 pts**) Consider the following C++ code sequence

**t = A[0];**

**for (i=0; i < 5; i++)**

**A[i] = A[i+1];**

**A[5] = t;**

Write an Assembly language program for MIPS processor, assuming that base address of the array **A** is in **$s0**.

**lw $t0, 0($s0) # t=A[0]**

**mov $t1, $zero # i=0**

**li $t4, 5 # $t4=5**

**loop:sll $t2, $t1, 2 # 4\*i**

**add $t2, $s0, $t2 # &A[i]**

**lw $t3, 4($t2) # $t3=A[i+1]**

**sw $t3, 0($t2) # #A[i]=$t3**

**addi $t1, $t1, 1 # i++**

**blt $t1, $t4, loop # check i<5**

**sw $t0, 20($s0) # A[5]=t**