Carleton University

Department of Systems and Computer Engineering

SYSC 4507 Winter 2017

Computer Systems Architecture

Assignment #1 Solution and Grading Scheme

Max marks: 100

1. **[50 marks]**
Suppose that a program is executed on a processor that has a clock frequency of **900** MHz and an average CPI of **1.6**.

a) [**25 marks**] If the program takes **1.8** seconds to execute, approximately how many instructions are in the program?

Clock = 900 MHz → period = 1/900 * 10^-6 seconds/cycle

1.6 cycles per instruction = 1.6 * 1/900 * 10^-6 seconds / instruction
= 16/9 * 10^-9 seconds/ instruction = 1.777 * 10^-9 seconds /instruction

1.8 seconds / (1.777 * 10^-9) seconds/instruction = 1.012943 * 10^9 instructions

Marking:
If showed correct final value without any calculations or derivation: 10 marks
If showed calculations or derivation (especially if final answer was incorrect):
5 marks for correct formula
5 marks for correct substitution
15 marks for correct final value:
Deduction of 2 for miswriting (eg. single number, exponent)

b) [**10 marks**] If the program takes 1.8 seconds to execute, what is the MIPS rating of the processor?

1.012943 * 10^9 instructions / 1.8 seconds = 0.562 * 10^9 instructions / second = 562 MIPS

Marking:
If incorrect final value:
Due to miswriting, but with correct approach otherwise: 7/10
Due to miscalculation, but with correct approach otherwise: 7/10

c) [**15 marks**]

(i) Performance depends on CPI, clock and number of instructions.
(ii) MIPS rating is not a measure of performance…

(iii) Compiling for a completely different processor may have a different ISA and may result in a different number of instructions.
(iv) Therefore, it is not possible to predict the percent speedup with the given information.

Marking:
7 marks for "can't predict speedup with given information", i.e. for including statement (iv)
8 marks for explaining that performance calculations for speedup requires knowledge of the number of instructions executed … which is not given.

2. **[50 Marks]**

Give the Big-O complexity for the following code fragment and briefly describe how you arrived at that expression.

```
y = 1;
sum = 0;
for ( r = 0 ;  r < n ; r++ )   // loop 1
{          y += 2 * y;
}
for ( j = 0 ;  j < n ;  j++ )   // loop 2
{          for ( k = j ;  k < n ;  k++ )     // loop 3
           {          for ( i = 0 ;  i < n ; i++ )   // loop 4
                      {          sum++;
                      }
           }
}
```

Loop 1: executes n times and completes before loop 2

# of times Loop 3 executes depends on j from loop 2:
1st time (j = 0): executes n times
2nd time (j = 1): executes n – 1 times
3rd time (j = 2): executes n – 2 times
Total number of executions = n + n – 1 + n – 2 + n – 3 + … 2 + 1 = n(n+1)/2 (Gauss' formula)
Loop 4: executes n times for each iteration of loop 3

Total number of executions of "sum++" in body of loop 4
= # loop 3 * # loop 4
= n(n+1)/2) * n = ((n^2 + n)/2) * n = (n^3 + n^2)/2
Include the loop 1: n + (n^3 + n^2)/2

O(n + (n^3 + n^2)/2) [remove constants and lower terms]
= O((n^3 + n^2)/2) -- remove lower order n, remove constant ½, remove lower order n^2
= O(n^3)

Therefore composed fragment is O(n^3)

Marking:

Correct Result : 25 marks O(n^3)

Correct Explanation : 25 marks

5 marks for correct iterations of loop 1

5 marks for correct use of Gauss' formula for iterations of loop 2 and loop 3

5 marks for correct iterations of loop 4

10 marks for correct derivation of overall complexity

(Note: the marking for the "weak" explanation is consistent with this mark breakdown)

Correct Result and "weak" explanation : 30 marks.

An example of "weak" explanation is something like:

We have three nested loops, the first two iterate n times and the third one iterates a maximum of n times thus the whole complexity is O(n*n*n) = O(n3).