OPTIMIZATION Of Compiler generated Code

Cs 342 Spring 2017

Take Home Test 3

Due by 11:59 PM , April 5, 2017

The **objective** of this assignment is to optimize compiler generated assembly code to clear an array(initialize all elements in the array to zero), and to compute inner product. In other words, to create a better assembly code than created by the compiler. You will need to measure performance (measure time) that it takes to compute a function for a problem of size ***N***. As an example we use a function to clear an array of size ***N***. This example for MIPS instructions is described in the textbook in a subsection on arrays versus pointers. You will need to perform performance measurements for the compiler generated assembly code to clear an array and then you optimize (create a better performance assembly code) the file and measure the performance again. You have to prove that your optimized code has a better performance.

This take home test project has three sections + Required CHALLENGE SECTION:

Section 1 Use Visual Studio environment for Parts A, B, C, D, E as described below.

Section 2 Use GCC in LINUX environment repeat Parts A, B, C, D, E as described below

Section 3 Use Inner product computation, (Instead of clear Array function) GCC in LINUX or Visual Studio environment and repeat Parts A, B, C, D, E for inner product computation.

Challenge section: Compute inner product computation using vector instructions,

And compare PERFORMANCE WITH SCALAR CODE.

Description of the take home test:

You will compare the performance of two programing approaches to clear an array of size ***N***.

Case 1: Using pointer arithmetic to access the array

Case 2: Use index arithmetic to access elements in the array

NOTE:

Use as a guide section in the text book describing these cases for MIPS processor. To understand better, I recommend you to run these example on MARS simulator. No performance measurements are required.

In your report you have to present a chart with the performance time versus the array size ***N***. ***N*** should range from 10, 100,1000,10000,100000, 1000000, 10000000. You should see performance gains with larger size of ***N***. It will be clear that for small sizes of ***N*** there will be no big performance difference.

For both Case **1, Case 2** complete the following parts:

OPTIMIZATION Of Compiler generated Code

Cs 342 Spring 2017

Take Home Test 3

Due by 11:59 PM , April 5, 2017

SECTION 1 MS Visual Studio environment

PART A

1. Create the main file in C/C++ and make sure that it compiles.
2. Create a separate file with the function code in C/C++ again and make sure that it compiles
3. Put the main file and the function file in a new project and make sure they compile separately.
4. Build the compiled files in your project and start a debugging session using single step execution.

PART B

1. Take the function file in C/C++ and generate an assembly listing file (.asm) for the function.
   Note: You want only assembly code listing for your file (no C/C++ source code, or machine instructions in your file).
2. To assemble or compile .asm files it is not a default project in Visual Studio environment, therefore you have to create a "Custom Build" for compiling .asm files, following the instructions given in class.
3. You will need to comment out some lines that create errors in the .asm until the function is compiled.

PART C

1. Create a new project that will contain the C/C++ file of main and the .asm file of the function that you just compiled in step B.3.
2. Make sure that both files compile, link ,build, and run in debug mode. Take notice that the main is in C/C++ and the function is in assembly in a separate file.
3. Save this project under the name "CompilerGeneratedFunction" index or pointer. You will use this project later as a basis for your experiment of performance measurement.

PART D

1. Make a copy of this project, exactly the same. Make sure that it builds and runs.
2. Analyze the compiler generated assembly code and optimize it as it was discussed in class.
3. Save this project under the name "OptimizedFunction".
4. Repeat parts C and D for clearing the array using index arithmetic and pointer arithmetic.

PART E

1. For performance measurements you need to create random static arrays of sizes N = 10, 100, 1000,10000, 100000, 1000000, 10000000.

OPTIMIZATION Of Compiler generated Code
Cs 342 Spring 2017
Take Home Test 3
Due by 11:59 PM , April 5, 2017

2. In each of the four cases you need to measure time and plot these measurements as a function of ***N***.

Please submit a report that includes all the points listed above. Explain what you are doing step by step using screenshots of you work.

SECTION 2 GCC in LINUX environment

Repeat Parts, A,B,C,D,E in LINUX environment for cases 1 and 2.

SECTION 3 Performance measurement of Inner product computation.

For INNER product computation *use expression*

$$(X, Y) = \sum_{i=0}^{N-1} x_i \, y_i$$

Where X, and Y are arrays of integers of size N.

Repeat Parts, A,B,C,D,E in LINUX or Visual studio environment.

OPTIMIZATION Of Compiler generated Code
Cs 342 Spring 2017
Take Home Test 3
Due by 11:59 PM , April 5, 2017

Appendix

Use the following code for your main file:

```
void main(int[], int);

static int  Array[10] ={1,2,3,4,5,6,7,8,9,-1};

int main() {
  int size = 10;
  ClearUsingIndexOptimized( Array, size);
}
```

Use the following code for the function to clear the array using indexes:

```
void ClearUsingIndex(int Array[], int size) {
int i;
for (i = 0; i < size; i +=1)
      Array[i] = 0;
}
```

Use the following code for the function to clear the array using indexes:

```
void ClearUsingPointers (int  *array, int size) {
int *p;
for (p = &array[0]; p < &array[size]; p = p + 1)
       *p = 0;
}
```

See details in the textbook, Chapter 2.15.

How to use the QueryPerformanceCounter function to time code in Visual C++

http://support.microsoft.com/kb/815668

```
// CodeTimer.cpp : Defines the entry point for the console application.
//Note You must add the common language runtime support compiler option
(/clr) in Visual C++ 2005 and up
//to successfully compile the  code sample.
//To add the common language runtime support compiler option in Visual C++
2005,
//follow these steps:
//a.Click Project, and then click <ProjectName> Properties.
//
// Note <ProjectName> is a placeholder for the name of the project.
// b.Expand Configuration Properties, and then click General.
// c.Click to select Common Language Runtime Support,  (/clr)
// in the Common Language Runtime support project setting in the right pane,
click Apply, and then click OK.

#include "stdafx.h"
#include <tchar.h>
#include <windows.h>
```

OPTIMIZATION Of Compiler generated Code

Cs 342 Spring 2017

Take Home Test 3

Due by 11:59 PM , April 5, 2017

```
using namespace System;

int _tmain(int argc, _TCHAR* argv[])
{
 __int64 ctr1 = 0, ctr2 = 0, freq = 0;
int acc = 0, i = 0;

// Start timing the code.
if (QueryPerformanceCounter((LARGE_INTEGER *)&ctr1)!= 0)
{
      // Code segment is being timed.
       for (i=0; i<65536; i++) acc++;

      // Finish timing the code.
      QueryPerformanceCounter((LARGE_INTEGER *)&ctr2);

      Console::WriteLine("Start Value: {0}",ctr1.ToString());
      Console::WriteLine("End   Value: {0}",ctr2.ToString());

      QueryPerformanceFrequency((LARGE_INTEGER *)&freq);
// freq  is number of counts per second. It approximates the CPU frequency
Console::WriteLine("QueryPerformanceFrequency : {0} counts per
Seconds.",freq.ToString());
//    Console::WriteLine(S"QueryPerformanceCounter minimum resolution: 1/{0}
Seconds.",freq.ToString());
    Console::WriteLine("QueryPerformanceCounter minimum resolution: 1/{0}
Seconds.",freq.ToString());
// In Visual Studio 2005, this line should be changed to:
Console::WriteLine("QueryPerformanceCounter minimum resolution: 1/{0}
Seconds.",freq.ToString());
Console::WriteLine("ctr2 - ctr1: {0} counts.",((ctr2 - ctr1) * 1.0 /
1.0).ToString());
      Console::WriteLine("65536 Increments by 1 computation time: {0}
seconds.",((ctr2 - ctr1) * 1.0 / freq).ToString());
}
else
{
      DWORD dwError = GetLastError();
    Console::WriteLine("Error value = {0}",dwError.ToString());
//    Console::WriteLine(S"Error value = {0}",dwError.ToString());// In
Visual Studio 2005, this line should be changed to: Console::WriteLine("Error
value = {0}",dwError.ToString());
}

// Make the console window wait.
Console::WriteLine();
Console::Write("Press ENTER to finish.");
Console::Read();


   return 0;
}
```