Last Name: First Name:

Computer Science
C.Sc. 342

Optimization of dot product computation of two vectors using vector instructions

Take Home TEST No. 3

CSc or CPE

Submit report and power point presentation < 2min by 12:00PM May 8, 2023

Please submit report, presentation to TA.

Please submit PPT to me for presentation in class.

(No extensions will granted for completing this test.)

Objective:

The objective of this take-home test is to optimize compiler generated code to compute dot product using vector instructions.

Tasks to perform:

1. *Use CPUID instruction to determine your processor vector processing capabilities.*
2. *Write C++ function to compute dot product in Visual Studio environment. Place the function in a separate file from main() that calls this function. Vector sizes should be powers of 2 (e.g. 16, 32, 64, ...512, ..2^{16} etc.). Disable Automatic Parallelization, /Qpar, and Automatic Vectorization, /arch.*
   Use QueryPerformanceCounter function to measure execution time.
   Plot graph: time versus vector size.
3. Compile code in §2. *Enable Automatic Parallelization, /Qpar, and Automatic Vectorization, /arch.*
   Use QueryPerformanceCounter function to measure execution time.
   Plot graph: time versus vector size.
   Inspect compiler generated assembly code. Observe if compiler vectorized code for very large vector sizes. Try to optimize compiler generated code. Based on compiler generated assembly code (or your optimized code) create an assembly code for dot product computation function (in the same way as shown in the text book for "clear-array example for MIPS"). Please refer to Tutorial in the Appendix.
   Use QueryPerformanceCounter function to measure execution time.
   Plot graph time versus vector size.
4. *To optimize the code further use vector instruction DPPS in function that computes dot product. instruction to Improve performance of vector assembly code in §2.* Use QueryPerformanceCounter function to measure execution time.
   Plot graph: time versus vector size.
5. Compare all plots in one figure.
6. Submit a detailed report and complete source code listing. If requested be ready to demo working project.
7. Perform this take home test in LINUX using gcc.
8. What to submit: 1. Write a report, 2.Create less than 2 min presentation using power point. 3. Source code files used in this project + Readme file with instructions.

DO NOT SUBMIT PROJECT FILES SHOWN IN TUTORIAL!

APPENDIX TUTORIAL STARTING ON NEXT PAGE

CSC 342, Spring 2023
Instructor Professor I.Gertner

TUTORIAL

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

http://support.microsoft.com/kb/815668
Exampled that worked.

```
// 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>

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;
}
```

CSC 342, Spring 2023
Instructor Professor I.Gertner

QueryPerformanceFrequency function

```
BOOL WINAPI QueryPerformanceFrequency(

  _Out_  LARGE_INTEGER *lpFrequency

);
```

Parameters

lpFrequency [out]

Type: **LARGE_INTEGER***

A pointer to a variable that receives the current performance-counter frequency, **in counts per second**. If the installed hardware does not support a high-resolution performance counter, this parameter can be zero.

Not related to CPU frequency in general
The high frequency counter need not be tied to the CPU frequency at all. It will only resemble the CPU frequency is the system actually uses the **TSC (TimeStampCounter)** underneath. As the **TSC is generally *unreliable* on multi-core systems it tends not to be used.** When the TSC is not used the ACPI Power Management Timer (pmtimer) may be used. You can tell if your system uses the ACPI PMT by checking if QueryPerformanceFrequency returns the signature value of 3,579,545 (ie 3.57MHz). If you see a value around 1.19Mhz then your system is using the old 8245 PIT chip. ***Otherwise you should see a value approximately that of your CPU frequency (modulo any speed throttling or power-management that might be in effect.)***

If you have a newer system with an invariant TSC (ie constant frequency TSC) then that is the frequency that will be returned (if Windows uses it). Again this is not necessarily the CPU frequency.

CSC 342, Spring 2023
Instructor Professor I.Gertner

Clear Array Using Indexs

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

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

int main()
{
  int size = 10;
// Start TIMER
  ClearUsingIndex( Array, size);
//STOP TIMER
// output the time difference stop_time-Star_time
}
```

Compiler generated code for procedure

```
// Clears array using indexing.
void ClearUsingIndex(int Array[], int size)
{
  int i;
  for (i = 0; i < size; i +=1)
        Array[i] = 0;
 }
```

```
; Listing generated by Microsoft (R) Optimizing Compiler Version
15.00.21022.08

      TITLE
      c:\Users\izidor64\Documents\CCNY_2012\Cs342\CS342Fall2012\Oct23_2012Cre
ateAssemblyfiles\ClearArrayIndex.cpp
      .686P
      .XMM
      include listing.inc
      .model      flat

INCLUDELIB MSVCRTD
INCLUDELIB OLDNAMES

PUBLIC      ?ClearUsingIndex@@YAXQAHH@Z               ; ClearUsingIndex
EXTRN __RTC_Shutdown:PROC
EXTRN __RTC_InitBase:PROC
;     COMDAT rtc$TMZ
; File
c:\users\izidor64\documents\ccny_2012\cs342\cs342fall2012\oct23_2012createass
emblyfiles\cleararrayindex.cpp
;rtc$TMZ    SEGMENT
;__RTC_Shutdown.rtc$TMZ DD FLAT:__RTC_Shutdown
;rtc$TMZ    ENDS
;     COMDAT rtc$IMZ
;rtc$IMZ    SEGMENT
```

```
;__RTC_InitBase.rtc$IMZ DD FLAT:__RTC_InitBase
; Function compile flags: /Odtp /RTCsu /ZI
;rtc$IMZ    ENDS
;     COMDAT ?ClearUsingIndex@@YAXQAHH@Z
_TEXT SEGMENT
_i$ = -8                                ; size = 4
_Array$ = 8                             ; size = 4
_size$ = 12                             ; size = 4
?ClearUsingIndex@@YAXQAHH@Z PROC                ; ClearUsingIndex, COMDAT
; Line 3
      push  ebp
      mov   ebp, esp
      sub   esp, 204                    ; 000000ccH
      push  ebx
      push  esi
      push  edi
      lea   edi, DWORD PTR [ebp-204]
      mov   ecx, 51                             ; 00000033H
      mov   eax, -858993460                     ; ccccccccH
      rep stosd
; Line 5
      mov   DWORD PTR _i$[ebp], 0                 ; i =0   on stack
      jmp   SHORT $LN3@ClearUsing
$LN2@ClearUsing:
      mov   eax, DWORD PTR _i$[ebp]         ; move again i from stack to eax
      add   eax, 1                              ; increament i in EAX
      mov   DWORD PTR _i$[ebp], eax             ; move eax onto  stack
$LN3@ClearUsing:
      mov   eax, DWORD PTR _i$[ebp]         ;  move i from stack to eax
      cmp   eax, DWORD PTR _size$[ebp] ; compare i in eax  with ARRAY  size
on stack
      jge   SHORT $LN4@ClearUsing               ; if done exit
; Line 6
      mov   eax, DWORD PTR _i$[ebp]             ; move again i into eax
      mov   ecx, DWORD PTR _Array$[ebp]         ; move address of the
ARRAY from stack to ecx
      mov   DWORD PTR [ecx+eax*4], 0  ; compute the effective address and
move zero to the address. This is the body of the loop
      jmp   SHORT $LN2@ClearUsing       ; jump to the begginning of the LOOP
$LN4@ClearUsing:
; Line 7
      pop   edi
      pop   esi
      pop   ebx
      mov   esp, ebp
      pop   ebp
      ret   0
?ClearUsingIndex@@YAXQAHH@Z ENDP                ; ClearUsingIndex
_TEXT ENDS
END
```

CSC 342, Spring 2023
Instructor Professor I.Gertner

Manually OPTIMIZED CODE

```
;
.686P
      .XMM
      include listing.inc
      .model      flat;
; Custom Build Step, including a listing file placed in intermediate
directory
; but without Source Browser information
; debug:
; ml -c -Zi "-Fl$(IntDir)\$(InputName).lst" "-
Fo$(IntDir)\$(InputName).obj" "$(InputPath)"
; release:
; ml -c "-Fl$(IntDir)\$(InputName).lst" "-Fo$(IntDir)\$(InputName).obj"
"$(InputPath)"
; outputs:
; $(IntDir)\$(InputName).obj

; Custom Build Step, including a listing file placed in intermediate
directory
; and Source Browser information also placed in intermediate directory
; debug:
; ml -c -Zi "-Fl$(IntDir)\$(InputName).lst" "-
FR$(IntDir)\$(InputName).sbr" "-Fo$(IntDir)\$(InputName).obj"
"$(InputPath)"
; release:
; ml -c "-Fl$(IntDir)\$(InputName).lst" "-FR$(IntDir)\$(InputName).sbr" "-
Fo$(IntDir)\$(InputName).obj" "$(InputPath)"
; outputs:
; $(IntDir)\$(InputName).obj
; $(IntDir)\$(InputName).sbr
```

```
PUBLIC     ?ClearUsingIndex@@YAXQAHH@Z            ; ClearUsingIndex

.code
_TEXT SEGMENT
_i$ = -8
_Array$ = 8
_size$ = 12
?ClearUsingIndex@@YAXQAHH@Z PROC            ; ClearUsingIndex, COMDAT
; Line 14
      push  ebp
      mov   ebp, esp
      sub   esp, 204                    ; 000000ccH
      push  ebx
      push  esi
      push  edi
      lea   edi, DWORD PTR [ebp-204]
      mov   ecx, 51                             ; 00000033H
      mov   eax, -858993460                     ; ccccccccH
      rep stosd
; Line 16
;Initialize:
      mov   eax, 0           ; initialize index i to 0 in Register
EAX
      mov   ecx, DWORD PTR _Array$[ebp]
      mov   edx, DWORD PTR _size$[ebp]
      jmp   SHORT $L281                         ;jump to Loop

$L282:
      add   eax, 1                              ;INCREMENT Index.
$L281:
      cmp   eax, edx                      ;Check index  < SIZE
      jge   SHORT $L279                         ;EXIT when DONE!
; Line 17
      mov   DWORD PTR [ecx+eax*4], 0            ; LOOP BODY!
      jmp   SHORT $L282                   ; control Loop.
                                          ;after removal
                                                      ;we are left
with 5 instruction
                                                      ; in LOOP!

$L279:
; Line 18
      pop   edi
      pop   esi
      pop   ebx
      mov   esp, ebp
      pop   ebp
      ret   0
?ClearUsingIndex@@YAXQAHH@Z ENDP            ; ClearUsingIndex
```

```
_TEXT ENDS
END
```

Clear Array Using Pointers

```
#define SIZE   10  /* number of integers in an Array */
void ClearUsingPointers(int *, int);

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

int main()
{
  //start timer
  ClearUsingPointers( &Array[0], SIZE);
//stop timer
//output time difference
}



Procedure Clears array using pointers.
void ClearUsingPointers(int *Array, int size)
{
  int *p;
  for (p = &Array[0]; p < &Array[size]; p = p+1)
        *p= 0;
};
```

Compiler generated code

```
; Listing generated by Microsoft (R) Optimizing Compiler Version
15.00.21022.08

      TITLE
      c:\Users\izidor64\Documents\CCNY_2012\Cs342\CS342Fall2012\Oct23_2012
CreateAssemblyFilesPOinters\ClearArrayPointer.cpp
      .686P
      .XMM
      include listing.inc
      .model      flat
; Custom Build Step, including a listing file placed in intermediate
directory
; but without Source Browser information
; debug:
; ml -c -Zi "-Fl$(IntDir)\$(InputName).lst" "-
Fo$(IntDir)\$(InputName).obj" "$(InputPath)"
; release:
; ml -c "-Fl$(IntDir)\$(InputName).lst" "-Fo$(IntDir)\$(InputName).obj"
"$(InputPath)"
; outputs:
; $(IntDir)\$(InputName).obj

; Custom Build Step, including a listing file placed in intermediate
directory
; and Source Browser information also placed in intermediate directory
; debug:
; ml -c -Zi "-Fl$(IntDir)\$(InputName).lst" "-
FR$(IntDir)\$(InputName).sbr" "-Fo$(IntDir)\$(InputName).obj"
"$(InputPath)"
; release:
; ml -c "-Fl$(IntDir)\$(InputName).lst" "-FR$(IntDir)\$(InputName).sbr" "-
Fo$(IntDir)\$(InputName).obj" "$(InputPath)"
; outputs:
; $(IntDir)\$(InputName).obj
; $(IntDir)\$(InputName).sbr

; Listing generated by Microsoft (R) Optimizing Compiler Version
15.00.21022.08

      TITLE
      c:\Users\izidor64\Documents\CCNY_2012\Cs342\CS342Fall2012\Oct23_2012
CreateAssemblyFilesPOinters\ClearArrayPointer.cpp
      .686P
      .XMM
      include listing.inc
      .model      flat

INCLUDELIB MSVCRTD
INCLUDELIB OLDNAMES

PUBLIC      ?ClearUsingPointers@@YAXPAHH@Z                ;
ClearUsingPointers
```

```
EXTRN __RTC_Shutdown:PROC
EXTRN __RTC_InitBase:PROC
;     COMDAT rtc$TMZ
; File
c:\users\izidor64\documents\ccny_2012\cs342\cs342fall2012\oct23_2012create
assemblyfilespointers\cleararraypointer.cpp
rtc$TMZ     SEGMENT
__RTC_Shutdown.rtc$TMZ DD FLAT:__RTC_Shutdown
rtc$TMZ     ENDS
;     COMDAT rtc$IMZ
rtc$IMZ     SEGMENT
__RTC_InitBase.rtc$IMZ DD FLAT:__RTC_InitBase
; Function compile flags: /Odtp /RTCsu /ZI
rtc$IMZ     ENDS
;     COMDAT ?ClearUsingPointers@@YAXPAHH@Z
_TEXT SEGMENT
_p$ = -8                                ; size = 4
_Array$ = 8                             ; size = 4
_size$ = 12                             ; size = 4
?ClearUsingPointers@@YAXPAHH@Z PROC                ; ClearUsingPointers,
COMDAT
; Line 5
      push  ebp
      mov   ebp, esp
      sub   esp, 204                    ; 000000ccH
      push  ebx
      push  esi
      push  edi
      lea   edi, DWORD PTR [ebp-204]
      mov   ecx, 51                             ; 00000033H
      mov   eax, -858993460                     ; ccccccccH
      rep stosd
; Line 7
      mov     eax, DWORD PTR _Array$[ebp]   ;Formal parameter to the Clear function ADDRESS of the array
      mov     DWORD PTR _p$[ebp], eax               ;local pointer to Array move to stack
      jmp     SHORT $LN3@ClearUsing
$LN2@ClearUsing:
      mov     eax, DWORD PTR _p$[ebp]               ;move outside of the LOOP to Initialize. DONE Line 17
      add     eax, 4                                ; increment pointer by 4
      mov     DWORD PTR _p$[ebp], eax               ; move incremented pointer back to stack
$LN3@ClearUsing:
      mov     eax, DWORD PTR _size$[ebp]            ;move outside Loop to load size of an array
      mov     ecx, DWORD PTR _Array$[ebp]           ;move outside of the LOOP
                                                     ;to Initialize      Address of Array
      lea     edx, DWORD PTR [ecx+eax*4]            ;move outside of the LOOP to Initialize to
                                                     ;the address of the last element in Array
      cmp     DWORD PTR _p$[ebp], edx
      jae     SHORT $LN4@ClearUsing                 ;EXIT if done
; Line 8
      mov     eax, DWORD PTR _p$[ebp]               ;remove. do not need it
      mov     DWORD PTR [eax], 0                    ; body of the loop mov 0 to the address in EAX
      jmp     SHORT $LN2@ClearUsing                 ; goto start of the LOOP
$LN4@ClearUsing:
; Line 9
      pop   edi
      pop   esi
```

```
      pop   ebx
      mov   esp, ebp
      pop   ebp
      ret   0
?ClearUsingPointers@@YAXPAHH@Z ENDP                ; ClearUsingPointers
_TEXT ENDS
END
```

POINTERS, Optimized Manually CODE

```
.386
.model flat, c

; Custom Build Step, including a listing file placed in intermediate
directory
; but without Source Browser information
; debug:
; ml -c -Zi "-Fl$(IntDir)\$(InputName).lst" "-
Fo$(IntDir)\$(InputName).obj" "$(InputPath)"
; release:
; ml -c "-Fl$(IntDir)\$(InputName).lst" "-Fo$(IntDir)\$(InputName).obj"
"$(InputPath)"
; outputs:
; $(IntDir)\$(InputName).obj

; Custom Build Step, including a listing file placed in intermediate
directory
; and Source Browser information also placed in intermediate directory
; debug:
; ml -c -Zi "-Fl$(IntDir)\$(InputName).lst" "-
FR$(IntDir)\$(InputName).sbr" "-Fo$(IntDir)\$(InputName).obj"
"$(InputPath)"
; release:
; ml -c "-Fl$(IntDir)\$(InputName).lst" "-FR$(IntDir)\$(InputName).sbr" "-
Fo$(IntDir)\$(InputName).obj" "$(InputPath)"
; outputs:
; $(IntDir)\$(InputName).obj
; $(IntDir)\$(InputName).sbr
```

```
.code
_TEXT SEGMENT
_p$ = -8
_Array$ = 8
_size$ = 12
ClearUsingPointers PROC NEAR      ; ClearUsingPointers, COMDAT
; Line 15
      push  ebp
      mov   ebp, esp
      sub   esp, 204                    ; 000000ccH
      push  ebx
      push  esi
      push  edi
      lea   edi, DWORD PTR [ebp-204]
      mov   ecx, 51                             ; 00000033H
      mov   eax, -858993460                     ; ccccccccH
      rep stosd
```

```
; INITIALIZATION outside of the LOOP!
; Line 17
      mov     eax, DWORD PTR _Array$[ebp]       ;Initialize Formal parameter to the Clear function
      mov     DWORD PTR _p$[ebp], eax           ;Reg EAX is a local pointer to an  Array
  mov ecx, DWORD PTR _Array$[ebp]               ;to Initialize      Address of an Array
  mov ebx, DWORD PTR _size$[ebp]                ;get SIZE to reg EBX
  lea edx, DWORD PTR [eax+ebx*4]                ;Initialize EDX to the address of the last element in Array
      jmp     SHORT $L280
$L281:                                          ; Beginning of LOOP
      add   eax, 4                              ;Increment Pointer by 4
$L280:
      cmp   eax, edx                            ; Compare TWO Registers if DONE
      jae   SHORT $L278                         ;EXIT the Loop if done
; Line 18
      mov   DWORD PTR [eax], 0                  ; Body of the Loop
      jmp   SHORT $L281                         ;  Go to Loop
                                                ;  we have 4 instructions in the Loop!
                                                ; no Effective address computation
$L278:
; Line 19
      pop   edi
      pop   esi
      pop   ebx
      mov   esp, ebp
```

```
      pop   ebp
      ret   0
ClearUsingPointers ENDP                ; ClearUsingPointers
_TEXT ENDS
END
```