Benchmark Cache Performance Analysis

Kyungchan Im

Grand Canyon University

Course number: CST - 307

Bill Hughes

Oct 2, 2022

Benchmark Cache Performance Analysis

In this documentation, it is divided into three sections: Assembly code, execution screenshot, and benchmark analysis. The requirement for this assignment is to check the miss penalty clock cycle time and average miss penalty for one word and four-word line size cache. Also, non-burst and burst transfer case for four-word cache line size are required to compare between the performance of non-burst and burst transfer. The assembly code given for this benchmark analyze and execute the calculation from the benchmark analysis part. The successful execution screenshot is given on the part two of this documentation, and finally the analysis part is listed on the part three.

MIPS Assembly Code

This MIPS assembly code is given from the class for this benchmark assignment. We modified and evaluated the .data value inside the file. The comments were edited for part of our assignment and calculated data.

----- Code Below Here ------

```
# Benchmark-Cache-Performance-Analysis.asm
#-------------------------------------------------
# Author: William B Hurst
# Creation date: 2019Sep18
#-------------------------------------------------
# this is a simple program that demonstrates
# how a computer system that utilizes 4-Byte Words
# can have significantly different Performance
# characteristics based on whether it uses
# Main Memory and Cache Data Transfer configurations of:
#
# 1) One Block Word
# 2) Four Block Words
# 3) Non-Burst
# 4) Burst
#
# from a computer system that utilizes:
#
# *)  4 byte words
#-------------------------------------------------
# the system has the following characteristics
#-------------------------------------------------
#       Observed Miss Penalty Rates of
#-------------------------------------------------
# *)  1 Clock Cycle to Read from Main Memory
# *)  3 clock Cycles to transfer the 4 blocks of data
#-------------------------------------------------
#
# The program asks the user to input:
#
# *) One  word block miss penalty rate
# *) Four word block miss penalty rate
#
#-------------------------------------------------
# the data will be stored as follows in the program:
#
#           $t0 - used to hold the first number
#           $t1 - used to hold the second number
#           $t2 - used to hold multiplication of $t0 * $t1
#-------------------------------------------------
# Declaration of .text and main
# Set main as global declaration
.text
.globl      main
main:
#-------------------------------------------------------------------------
li $v0, 4                       # Prepare syscall for print
la $a0, programDesc             # assign address value on $a0 --> ProgramDesc
syscall                         # syscall print
#-------------------------------------------------------------------------
li $v0, 4                       # Prepare syscall for print
la $a0, oWMissRateReq           # assign address value on $a0 --> oWMissRateReq
syscall                         # syscall print
#-------------------------------------------------------------------------
li $v0, 6                       # Prepare syscall for reading a float value |code 6 = reading a float
```

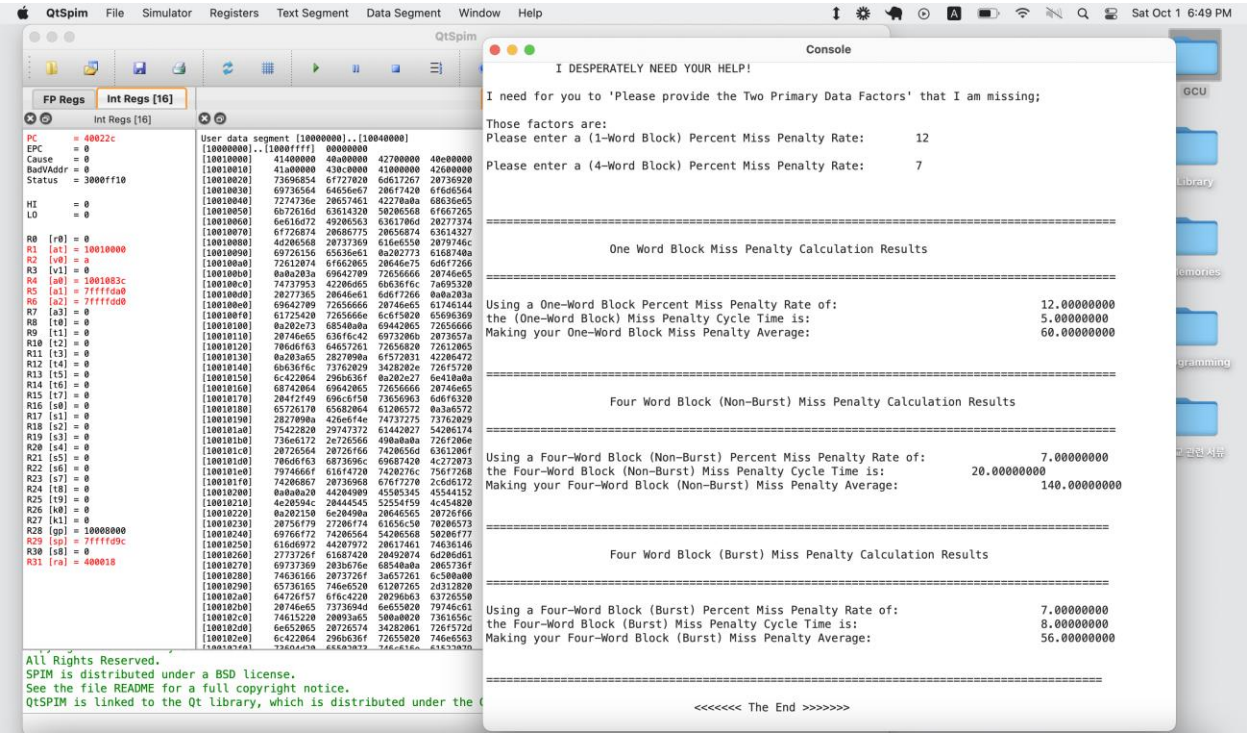
```
syscall                         # syscall reading a float
#---------------------------------------------------------------------
s.s $f0, oWordMissRate          # # store word from $f0 into address oWordMissRate
#---------------------------------------------------------------------
li $v0, 4                       # Prepare syscall for print
la $a0, fWMissRateReq           # assign address value on $a0 --> fwMissRateReq
syscall
#---------------------------------------------------------------------
li $v0, 6                       # Prepare syscall for reading a float value
syscall                         # syscall
#---------------------------------------------------------------------
 s.s $f0, fWordMissRate         # store word from $f0 into address fWordMissRate
#---------------------------------------------------------------------
# perform the intended operation - multiply
#  Miss_Penalty_ave = (Miss_Penalty_rate)(Miss_Penalty_Cycles)
#---------------------------------------------------------------------
# One Word Miss Penalty Calculations
#---------------------------------------------------------------------
l.s $f0, oWordMissRate          # load word into $f0 from address oWordMissRate
l.s $f1, oWordMissCTime         # load word into $f1 from address oWordMissTime
mul.s $f2, $f0, $f1             # Multiply two float point and result to one.
s.s $f2, oWordMissAve           # store word from $f2 into address oWordMissAve
#---------------------------------------------------------------------
li $v0, 4                       # Prepare syscall for print
la $a0, oWordMissCalcOut        # assign address value on $a0 --> oWordMissCalcOut
syscall                         # Execute the syscall
#---------------------------------------------------------------------
li $v0, 4                       # Same step follows
la $a0, oWordMissRateOut
syscall
#---------------------------------------------------------------------
li $v0, 2                       # Prepare syscall for print float | code 2 = print float
l.s $f12, oWordMissRate         # load float into $f12 from address oWordMissRate
syscall                         # Execute
#---------------------------------------------------------------------
li $v0, 4                       # Prepare printing oWordMissCTimeOut
la $a0, oWordMissCTimeOut
syscall
#---------------------------------------------------------------------
li $v0, 2                       # Prepare syscall for printing float
l.s $f12, oWordMissCTime        # load float into $f12 from address oWordMissCTime
syscall
#---------------------------------------------------------------------
li $v0, 4                       # Same process code 4 = printing string
la $a0, oWordMissAveOut
syscall
#---------------------------------------------------------------------
li $v0, 2                       # Process for printing float stored in the oWordMissAve address
l.s $f12, oWordMissAve
syscall
#---------------------------------------------------------------------
# Four Word (Non-Burst) Miss Penalty Calculations
#---------------------------------------------------------------------
l.s $f0, fWordMissRate          # Store the result of multiplied float value into nbfWordMissAve($f2)
l.s $f1, nbfWordMissCTime       # Multiplication = fWordMissRate * nbfWordMissCTime
mul.s       $f2, $f0, $f1
s.s         $f2, nbfWordMissAve
#---------------------------------------------------------------------
li $v0, 4                       # Print the string stored in the nbfWordMissCalcOut
la $a0, nbfWordMissCalcOut
syscall
#---------------------------------------------------------------------
li $v0, 4                       # Print the string stored in the nbfWordMissRateOut
la $a0, nbfWordMissRateOut
syscall
#---------------------------------------------------------------------
li $v0, 2                       # Print the float stored in fWordMissRate
l.s $f12, fWordMissRate
syscall
#---------------------------------------------------------------------
```

```
li $v0, 4                       # Print the string stored in the nbfWordMissCTimeOut
la $a0, nbfWordMissCTimeOut
syscall
#---------------------------------------------------------------------
li $v0, 2                       # Print the float stored in nbfWordMissCTime
l.s $f12, nbfWordMissCTime
syscall
#---------------------------------------------------------------------
li $v0, 4                       # Print the string stored in nbfWordMissAveOut
la $a0, nbfWordMissAveOut
syscall
#---------------------------------------------------------------------
li $v0, 2                       # Print the float stored in nbfWordMissAve
l.s $f12, nbfWordMissAve
syscall
#---------------------------------------------------------------------
# Four Word (Burst) Miss Penalty Calculations
#---------------------------------------------------------------------
l.s $f0, fWordMissRate          # Store the result of multiplied float value into bfWordMissAve($f2)
l.s $f1, bfWordMissCTime        # Multiplication = fWordMissRate * bfWordMissCTime
mul.s $f2, $f0, $f1
s.s $f2, bfWordMissAve
#---------------------------------------------------------------------
li $v0, 4                       # Print the string stored in bfWordMissCalcOut address
la $a0, bfWordMissCalcOut
syscall
#---------------------------------------------------------------------
li $v0, 4                       # Print the string stored in bfWordMissRateOut address
la $a0, bfWordMissRateOut
syscall
#---------------------------------------------------------------------
li $v0, 2                       # Print the float stored in fWordMissRate address
l.s $f12, fWordMissRate
syscall
#---------------------------------------------------------------------
li $v0, 4                       # Print the string stored in bfWordMissCTimeOut address
la $a0, bfWordMissCTimeOut
syscall
#---------------------------------------------------------------------
li $v0, 2                       # Print the float stored in bfWordMissCTime address
l.s $f12, bfWordMissCTime
syscall
#---------------------------------------------------------------------
li $v0, 4                       # Print the string stored in bfWordMissAveOut address
la $a0, bfWordMissAveOut
syscall
#---------------------------------------------------------------------
li $v0, 2                       # Print the float stored in bfWordMissAve address
l.s $f12, bfWordMissAve
syscall
#---------------------------------------------------------------------
li $v0, 4                       # PRint the string stored in theEnd address
la $a0, theEnd
syscall
#---------------------------------------------------------------------
li $v0, 10                      # Prepare syscall for terminate the program | code 10 = Exit
syscall                         # Execute
#---------------------------------------------------------------------
        .data
#-----------------------------------
oWordMissRate:          .float  0.12
oWordMissCTime:         .float  5.0             # place answer to a here.
oWordMissAve:           .float  0.6
#-----------------------------------
fWordMissRate:          .float  0.07
#-----------------------------------
nbfWordMissCTime: .float        20.0            # place answer to b here.
nbfWordMissAve:         .float  1.4
#-----------------------------------
bfWordMissCTime:   .float       8.0             # place answer to c here.
```

```
bfWordMissAve:                  .float      0.56
#-------------------------------------------------------------------------
programDesc:                    .asciiz "This program is designed to demonstrate
\n'Benchmark Cache Performance Impacts' through the 'Cache Miss Penalty Variances'
\nthat are found from:
\n\t'different System Block Sizes' and from:
\n\t'different Data Tranfer Policies.
\n\nThe Different Block sizes compared here are:
\n\t'(1 Word Block) vs. (4 Word Block)'.
\n\nAnd the different I/O Policies compared here are:
\n\t'(NonBurst) vs (Burst)' Data Transfer.
\n\nIn order for me to accomplish this 'Lofty Goal' through this program,
\n\n\tI DESPERATELY NEED YOUR HELP!
\nI need for you to 'Please provide the Two Primary Data Factors' that I am missing;
\nThose factors are:"
oWMissRateReq:      .asciiz     "\nPlease enter a (1-Word Block) Percent Miss Penalty Rate:\t  "
fWMissRateReq:      .asciiz     "\nPlease enter a (4-Word Block) Percent Miss Penalty Rate:\t  "
#-------------------------------------------------------------------------
oWordMissCalcOut: .asciiz "\n\n
==========================================================================================\n
\t      One Word Block Miss Penalty Calculation Results\n
==========================================================================================\n"
#-------------------------------------------------------------------------
oWordMissRateOut:  .asciiz      "\nUsing a One-Word Block Percent Miss Penalty Rate of: \t\t\t"
oWordMissCTimeOut:              .asciiz "\nthe (One-Word Block) Miss Penalty Cycle Time is: \t\t\t\t"
oWordMissAveOut:   .asciiz "\nMaking your One-Word Block Miss Penalty Average: \t\t\t\t"
#-------------------------------------------------------------------------
nbfWordMissCalcOut: .asciiz "\n\n
==========================================================================================\n
\t      Four Word Block (Non-Burst) Miss Penalty Calculation Results\n
==========================================================================================\n"
#-------------------------------------------------------------------------
nbfWordMissRateOut:             .asciiz     "\nUsing a Four-Word Block (Non-Burst) Percent Miss Penalty Rate of: \t\t"
nbfWordMissCTimeOut:            .asciiz "\nthe Four-Word Block (Non-Burst) Miss Penalty Cycle Time is: \t\t"
nbfWordMissAveOut:.asciiz "\nMaking your Four-Word Block (Non-Burst) Miss Penalty Average: \t\t"
#-------------------------------------------------------------------------
bfWordMissCalcOut: .asciiz "\n\n
==========================================================================================\n
\t      Four Word Block (Burst) Miss Penalty Calculation Results\n
==========================================================================================\n"
#-------------------------------------------------------------------------
bfWordMissRateOut: .asciiz      "\nUsing a Four-Word Block (Burst) Percent Miss Penalty Rate of: \t\t"
bfWordMissCTimeOut:             .asciiz "\nthe Four-Word Block (Burst) Miss Penalty Cycle Time is: \t\t\t"
bfWordMissAveOut:  .asciiz "\nMaking your Four-Word Block (Burst) Miss Penalty Average: \t\t\t"
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
cflf:   .asciiz             "\n"
theEnd: .asciiz "\n\n
==========================================================================================\n
\t\t\t<<<<<<< The End >>>>>>>\n
==========================================================================================\n"
```

Program Execution Screenshot

This is the screenshot of the program running on QtSpim. The evaluation and result are listed on the console. The calculations and given data are listed below this documentation.



Benchmark Analysis

With the assembly program “Benchmark-Cache-Performance-Analysis.asm”, you now have a program that calculates ‘Cache Miss Penalty Variances’

based on:

(1-Word Block Size) vs (4-Word Block Size)
and

(NonBurst) vs (Burst) Data Transfer

The calculations for the $Miss\_Penalty_{Clock\_Cycle\_Times(CCT)}$ are as follows:

Assume the following performance characteristics on a cache read miss: one clock cycle to access cache memory, one clock cycle to access main memory, and six clock cycles to transfer a 32-bit word to the processor and cache.

a. If the cache line size is one word, what is the miss penalty (i.e., additional time required for a read in the event of a read miss)?

For one word line size cache, there will be 5 clock cycles

Miss Penalty$_{Clock_Cycle_Times}$ = cache line size * (Clock Cycle for sending address to main memory + clock cycle for transfer word)

Miss Penalty$_{Clock_Cycle_Times}$ = 1 * (Clock Cycle for sending address to main memory + clock cycle for transfer word)

= 1 * (1 + 4)

= 5

b. What is the miss penalty if the cache line size is four words and a multiple, nonburst transfer is executed?

For four word line size cache, there will be 20 clock cycles when it is in non-burst transfer.

Miss Penalty$_{Clock_Cycle_Times}$ = (total line size cache) * (Clock Cycle for sending address to main memory + transfer word)

Miss Penalty$_{Clock_Cycle_Times}$ = (4, since there's four word) * (5, since one word required 5 total)

= 4 * 5

= 20

c. What is the miss penalty if the cache line size is four words and a transfer is executed, with one clock cycle per word transfer?

For four word line size cache, there will be 8 clock cycle when it is burst transfer.

Miss Penalty$_{Clock_Cycle_Times}$ = (One clock cycle for send an address to main memory) + (Four clock cycle to access 32-bit word from main memory) + (Three clock cycle for accessing remaining words in main memory = 3 remain)

Miss Penalty$_{Clock_Cycle_Times}$ = 1 + 4 + 3

Miss Penalty$_{Clock_Cycle_Times}$ = 8

Your new assignment is to modify the "Benchmark-Cache-Performance-Analysis.asm" program so that it will fulfill these parameters:

You have a cache read miss penalty:

- 1 clock cycle to access an address (cache and main memory), and 6 clock cycles to transfer the data

You will also have to calculate the results using the following Miss Penalty Rates

- 1-block size Miss Penalty Rate of 12%
- 4-block size Miss Penalty Rate of 7%

Make the changes to the program, you have been provided, to conform to the requirements as they are listed above.

Put your Final Answers here:

1) 1-Block Size:…….. 0.6 clock cycles
   a) Average Miss Penalty = One Block Miss Penalty Rate * Miss Penalty Time
      = 0.12 * 5
      = 0.6 clock cycles
2) 4-Block Size (nonBurst):……1.4 clock cycles
   a) Average Miss Penalty = Four Block Miss Penalty Rate * Non-Burst Miss Penalty Time
      = 0.07 * 20
      = 1.4 clock cycles
3) 4-Block Size (Burst):……..0.56 clock cycles
   a) Average Miss Penalty = Four Block Miss Penalty Rate * Burst Miss Penalty Time
      = 0.07 * 8
      = 0.56 clock cycles