CSE-420
Project-2_Report

1) Problem 1
   a. Question-1
      i. Statistics for the benchmarks are enlisted in **filename: q1_stats_table**

      ii. Non constant statistics between two simulation modes are as following :
      # Number of ticks simulated
      # Number of cpu cycles simulated
      # Number of busy cycles
      These are highlighted red in filename: q1_stats_table.

   b. Question-2
   From the stats file we can observe that Number of ticks simulated, Number of cpu cycles simulated, Number of busy cycles of timing model are approx. 100 times that of atomic model. Hence, we can say that atomic model has higher speed compared to a timing model.

2) Problem 2
   a. Question-1
      i. CPU model selected : Timing

      Explanation: Even though atomic model is faster than timing but timing model is more detailed CPU modelling. So the accuracy of timing model is more compared to atomic. In case of atomic model, Memory accesses are atomic while in case of timing, CPU waits until memory access returns, i.e., they use timing path. By choosing timing model we can observe and analyse the accurate program statistics more effectively than atomic model.

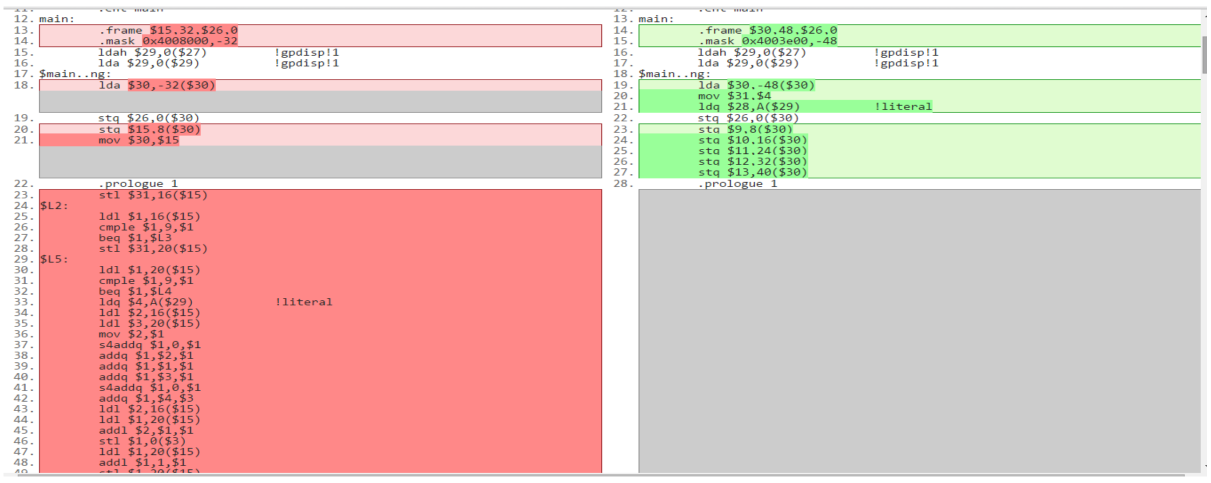
      ii. Working binary and assembly files for optimization with –O0 and –O3.
      Folder name: final q2-1

      iii. Comparison:
      Folder name: final q2-1
      Filename: O0-O3_stats_comparison_optimization

      - Number of instructions to be simulated and number of cpu cycles are less for –O3 optimization.
      - Number of integer registers read/write is lesser in –O3 optimization.
      - Number of load/store instructions are reduced.

      Hence, Better optimization is achieved by –O3 compared to –O0 in terms of speed and time.

Original vs –O3>

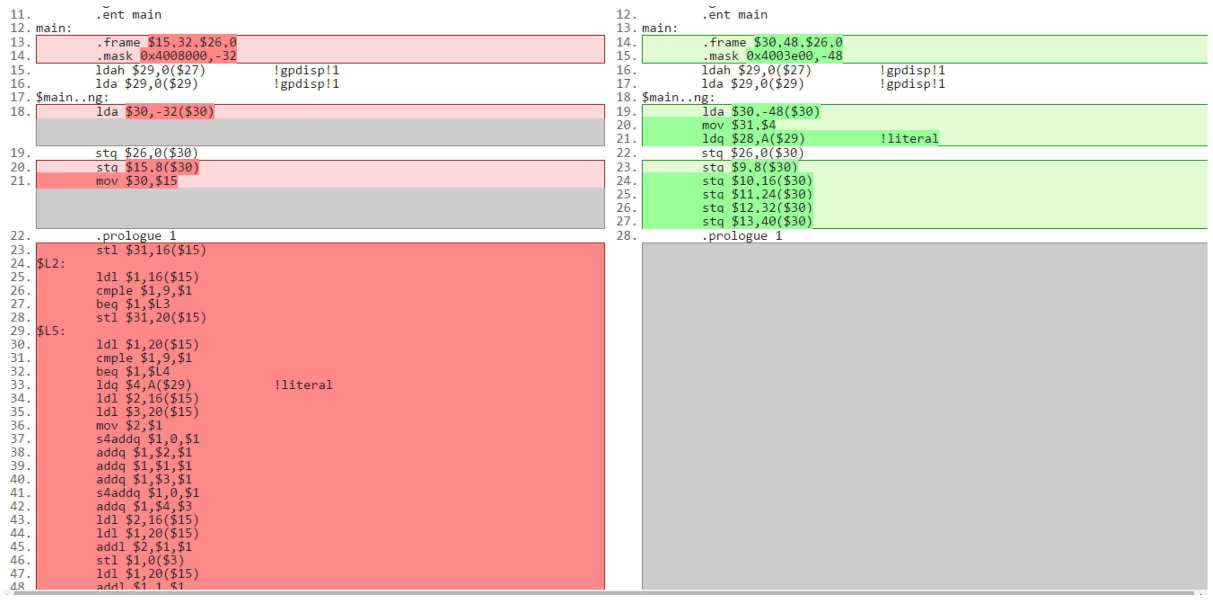


```
main:
        .frame $15,32,$26,0
        .mask 0x4008000,-32
        ldah $29,0($27)         !gpdisp!1
        lda $29,0($29)          !gpdisp!1
$main..ng:
        lda $30,-32($30)
        stq $26,0($30)
        stq $15,8($30)
        mov $30,$15
        .prologue 1
        stl $31,16($15)
$L2:
        ldl $1,16($15)
        cmple $1,9,$1
        beq $1,$L3
        stl $31,20($15)
$L5:
        ldl $1,20($15)
        cmple $1,9,$1
        beq $1,$L4
        ldq $4,A($29)           !literal
        ldl $2,16($15)
        ldl $3,20($15)
        mov $2,$1
        s4addq $1,0,$1
        addq $1,$2,$1
        addq $1,$1,$1
        addq $1,$3,$1
        s4addq $1,0,$1
        addq $1,$4,$3
        ldl $2,16($15)
        ldl $1,20($15)
        addl $2,$1,$1
        stl $1,0($3)
        ldl $1,20($15)
        addl $1,1,$1
```

```
main:
        .frame $30,48,$26,0
        .mask 0x4003e00,-48
        ldah $29,0($27)         !gpdisp!1
        lda $29,0($29)          !gpdisp!1
$main..ng:
        lda $30,-48($30)
        mov $31,$4
        ldq $28,A($29)          !literal
        stq $26,0($30)
        stq $9,8($30)
        stq $10,16($30)
        stq $11,24($30)
        stq $12,32($30)
        stq $13,40($30)
        .prologue 1
```

The label L20 has redundant code which is repeated and hence removed completely after optimizations which reduces code length and improves characteristics of model.

Original vs –O0>



```
        .ent main
main:
        .frame $15,32,$26,0
        .mask 0x4008000,-32
        ldah $29,0($27)         !gpdisp!1
        lda $29,0($29)          !gpdisp!1
$main..ng:
        lda $30,-32($30)
        stq $26,0($30)
        stq $15,8($30)
        mov $30,$15
        .prologue 1
        stl $31,16($15)
$L2:
        ldl $1,16($15)
        cmple $1,9,$1
        beq $1,$L3
        stl $31,20($15)
$L5:
        ldl $1,20($15)
        cmple $1,9,$1
        beq $1,$L4
        ldq $4,A($29)           !literal
        ldl $2,16($15)
        ldl $3,20($15)
        mov $2,$1
        s4addq $1,0,$1
        addq $1,$2,$1
        addq $1,$1,$1
        addq $1,$3,$1
        s4addq $1,0,$1
        addq $1,$4,$3
        ldl $2,16($15)
        ldl $1,20($15)
        addl $2,$1,$1
        stl $1,0($3)
        ldl $1,20($15)
```

```
        .ent main
main:
        .frame $30,48,$26,0
        .mask 0x4003e00,-48
        ldah $29,0($27)         !gpdisp!1
        lda $29,0($29)          !gpdisp!1
$main..ng:
        lda $30,-48($30)
        mov $31,$4
        ldq $28,A($29)          !literal
        stq $26,0($30)
        stq $9,8($30)
        stq $10,16($30)
        stq $11,24($30)
        stq $12,32($30)
        stq $13,40($30)
        .prologue 1
```

The label L20 has redundant code which is repeated and hence removed partially (1/2 of the iterations) after optimizations which reduces code length and improves characteristics of model.

Explanation:
No. of lines of code in assembly of original= 410
No. of lines of code with O0 optimization in assembly= 217
No. of lines of code with O3 optimization in assembly = 131

-o = The compiler tries to reduce the code size and execution time without performing any optimization that take a great deal of compilation time.

-O0= Reduced compilation time and make debugging produced the expected results.

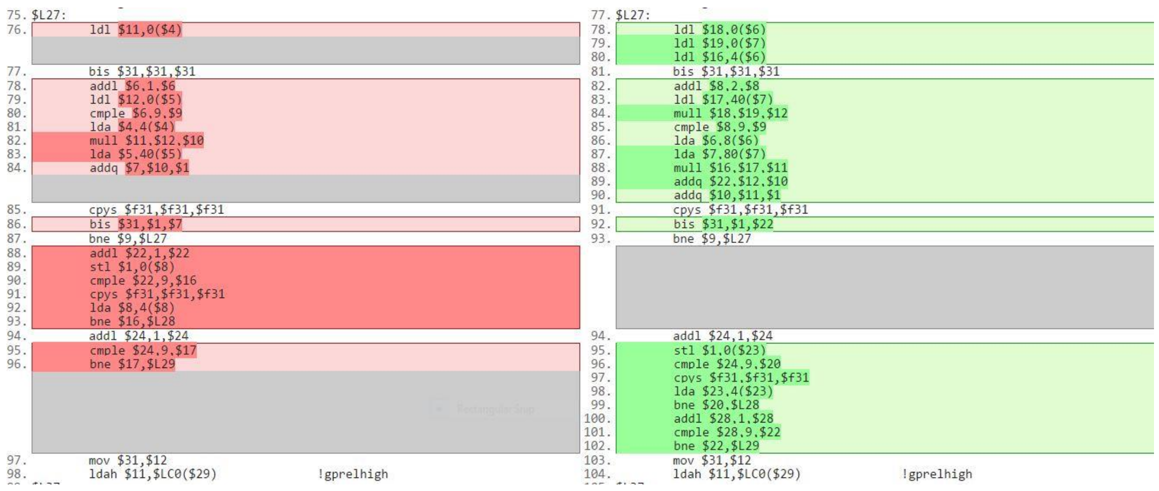
-O3= Optimize even more.
Hence, we observe that with optimization the length of the assembly code decreases.

b. Question-2
   i. Working unrolled files
      Folder name: final_q2-2

   ii. Comparison:
      Folder name: final_q2-2
      Filename: unrolling_stats_comparison

      We observed that number of conditional control instructions decreases as the level of unrolling increases. Hence, conditional branch instruction is executed lesser times. As we unroll the k-loop, the length of the code increases and the execution time goes on increasing with more unrolling.

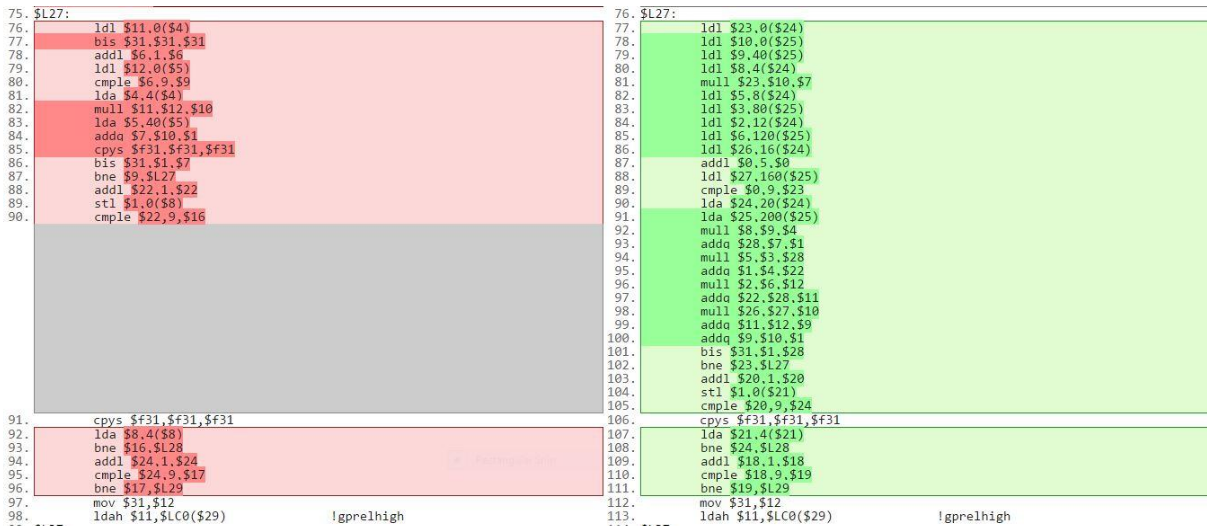
Original vs unrolling 2-times>



```
$L27:
        ldl $11,0($4)
        bis $31,$31,$31
        addl $6,1,$6
        ldl $12,0($5)
        cmple $6,9,$9
        lda $4,4($4)
        mull $11,$12,$10
        lda $5,40($5)
        addq $7,$10,$1
        cpys $f31,$f31,$f31
        bis $31,$1,$7
        bne $9,$L27
        addl $22,1,$22
        stl $1,0($8)
        cmple $22,9,$16
        cpys $f31,$f31,$f31
        lda $8,4($8)
        bne $16,$L28
        addl $24,1,$24
        cmple $24,9,$17
        bne $17,$L29
        mov $31,$12
        ldah $11,$LC0($29)              !gprelhigh
```

```
$L27:
        ldl $18,0($6)
        ldl $19,0($7)
        ldl $16,4($6)
        bis $31,$31,$31
        addl $8,2,$8
        ldl $17,40($7)
        mull $18,$19,$12
        cmple $8,9,$9
        lda $6,8($6)
        lda $7,80($7)
        mull $16,$17,$11
        addq $22,$12,$10
        addq $10,$11,$1
        cpys $f31,$f31,$f31
        bis $31,$1,$22
        bne $9,$L27
        addl $24,1,$24
        stl $1,0($23)
        cmple $24,9,$20
        cpys $f31,$f31,$f31
        lda $23,4($23)
        bne $20,$L28
        addl $28,1,$28
        cmple $28,9,$22
        bne $22,$L29
        mov $31,$12
        ldah $11,$LC0($29)              !gprelhigh
```

Original vs unrolling 5-times>



```
$L27:
        ldl $11,0($4)
        bis $31,$31,$31
        addl $6,1,$6
        ldl $12,0($5)
        cmple $6,9,$9
        lda $4,4($4)
        mull $11,$12,$10
        lda $5,40($5)
        addq $7,$10,$1
        cpys $f31,$f31,$f31
        bis $31,$1,$7
        bne $9,$L27
        addl $22,1,$22
        stl $1,0($8)
        cmple $22,9,$16
        cpys $f31,$f31,$f31
        lda $8,4($8)
        bne $16,$L28
        addl $24,1,$24
        cmple $24,9,$17
        bne $17,$L29
        mov $31,$12
        ldah $11,$LC0($29)              !gprelhigh
```

```
$L27:
        ldl $23,0($24)
        ldl $10,0($25)
        ldl $9,40($25)
        ldl $8,4($24)
        mull $23,$10,$7
        ldl $5,8($24)
        ldl $3,80($25)
        ldl $2,12($24)
        ldl $6,120($25)
        ldl $26,16($24)
        addl $0,5,$0
        ldl $27,160($25)
        cmple $0,9,$23
        lda $24,20($24)
        lda $25,200($25)
        mull $8,$9,$4
        addq $28,$7,$1
        mull $5,$3,$28
        addq $1,$4,$22
        mull $2,$6,$12
        addq $22,$28,$11
        mull $26,$27,$10
        addq $11,$12,$9
        addq $9,$10,$1
        bis $31,$1,$28
        bne $23,$L27
        addl $20,1,$20
        stl $1,0($21)
        cmple $20,9,$24
        cpys $f31,$f31,$f31
        lda $21,4($21)
        bne $24,$L28
        addl $18,1,$18
        cmple $18,9,$19
        bne $19,$L29
        mov $31,$12
        ldah $11,$LC0($29)              !gprelhigh
```

The length of the L27 loop is increased as no of instructions in branch loop is increased so branch is executing lesser times.

<u>Explanation:</u>
Therefore, we can say that more we unroll the loop lesser will be cycles per instruction, i.e. CPI. Because this rescheduled and unrolled loop program will hit branch instruction lesser times compared to original one. More the unrolling lesser times it will come to execute branch instruction.

When we unroll the k-loop 2-times it will reduce overall stalls for that loop. The overall stalls can further be reduced using more unrolling so is the case with unrolling 5-times. But again it will be limited by no. of registers we have.