

コンピューター・アーキテクチャレポート課題

1w152354-4 真殿航輝

課題出題日 2019/11/15

課題提出日 2019/11/20

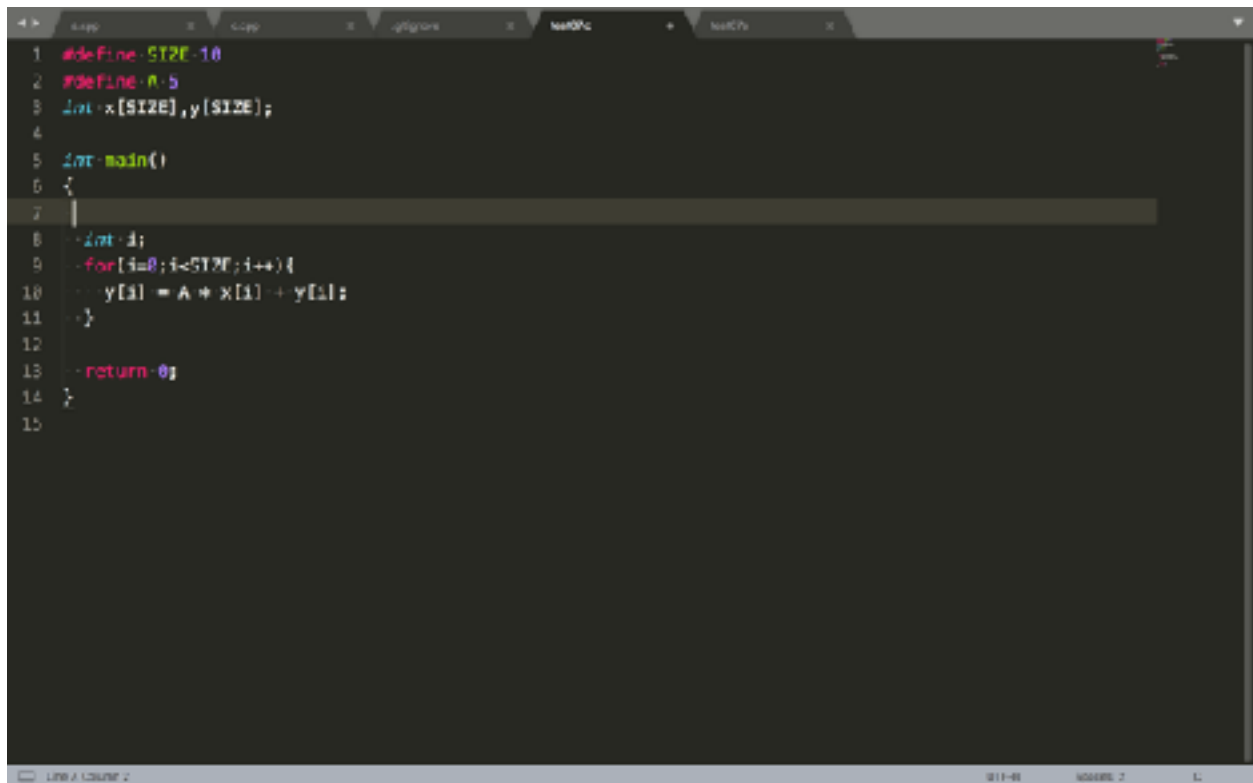
1.報告事項と考察

①アセンブリファイルを生成し、その中のループ本体部分を提示。どのようにループ本体部分を特定したか根拠の説明

初めにアセンブリファイルを

mipsel-linux-gnu-gcc -S test07.c

のコマンドを使用して作成した。test07.cには下の図1のように作成してある。今回は学籍番号の下1桁が5だったのでAを5として扱っている。このコマンドによって生成したファイル(test07.s)を図2に掲載する。



```
1 #define SIZE 10
2 #define N 5
3 int x[SIZE], y[SIZE];
4
5 int main()
6 {
7
8     int i;
9     for(i=0; i<SIZE; i++){
10         y[i] = A * x[i] + y[i];
11     }
12
13     return 0;
14 }
15
```

図1 test07.c

```

.file          1 "test07.c"
.section .mdebug.abi32
.previous
.abicalls
.text
.align        2
.globl        main
.ent          main
.type         main, @function

main:
.frame        $fp,24,$31           # vars= 8, regs= 1/0, args= 0, gp= 8
.mask         0x40000000,-8
.fmask        0x00000000,0
.set          noreorder
.cpload       $25
.set          nomacro

        addiu   $sp,$sp,-24
        sw      $fp,16($sp)
        move    $fp,$sp
        sw      $0,8($fp)
        b       $L2
        nop

$L3:
        lw      $5,8($fp)
        lw      $2,8($fp)
        lw      $3,%got(x)($28)
        sll     $2,$2,2
        addu    $2,$2,$3
        lw      $3,0($2)
        nop
        move    $2,$3
        sll     $2,$2,2
        addu    $4,$2,$3
        lw      $2,8($fp)
        lw      $3,%got(y)($28)
        sll     $2,$2,2
        addu    $2,$2,$3
        lw      $2,0($2)
        nop
        addu    $4,$4,$2
        lw      $3,%got(y)($28)
        sll     $2,$2,2
        addu    $2,$2,$3
        sw      $4,0($2)
        lw      $2,8($fp)
        nop
        addiu   $2,$2,1
        sw      $2,8($fp)

$L2:
        lw      $2,8($fp)
        nop
        slt     $2,$2,10
        bne     $2,$0,$L3
        nop

        move    $2,$0
        move    $sp,$fp
        lw      $fp,16($sp)
        addiu   $sp,$sp,24
        j       $31
        nop

        .set    macro
        .set    reorder
        .end

.comm          x,40,4

.comm          y,40,4
.ident         "GCC: (GNU) 4.2.4"

```

図2 test07.s ファイルの内容

ループ内は\$L3命令で\$L2でそのループ命令をループさせるかの条件文命令を実行している。こう考えた理由は、slt はオペランドが[Rd, Rs, Rt]であった時にRd = if Rs < Rt then 1 else 0という命令を実行して、bneはオペランドが[Rs, Rt, label]であった時にgoto label if Rs != Rtを実行するために、main内で\$0が0に初期化されて、毎回\$fp+8のアドレスに格納された\$L3での計算結果が\$2にloadされて、\$2が10を超える値になっった時に\$2に0が入るようになっているからである。\$L3内でaddiu \$2,\$2,1で\$2の値をインクリメントしているのも根拠の一つである。故に\$L3がループ箇所である。

②コンパイルして生成された実行バイナリを逆アセンブルし、その中のループ 本体部分を提示

```
400808: 00521021 addu v0,v0,s2
40080c: 8c590000 lw t9,0(v0)
400810: 0320f809 jalr t9
400814: 26100001 addiu s0,s0,1
400818: 8fbc0010 lw gp,16(sp)
40081c: 0211102b sltu v0,s0,s1
400820: 1440fff9 bnez v0,400808 <__uClibc_main+0x250>
```

図3 逆アセンブル結果のループ箇所について

根拠としてはaadiuでs0が1だけインクリメントされていて、その後にsltu,bnez命令でfor分内で分岐命令が走っているためである。分岐先に400808が指定されているので、そこに飛んだ後に復帰アドレスとして「400810」が格納されているので計算後に返ってきて、というのを繰り返している。

③実行バイナリをSimPipeで実行し、クロックサイクル数とパイプのログを示す。実行はフォワーディング有りと無しの両方で行う

フォワーディング無しの場合

* Pipeline Configuration *

Forwarding: No

DataCache Disabled

#####

cycle count: 1366

inst count: 786

IPC: 0.575403

simulation time: 0.001

パイプのログをループ本体部分の1ループ分の対応箇所

```
55: | 4005f0|  lw|  sw|  sw|  sw|  sw|
    |
56: | 4005f4|  lw|  lw|  sw|  sw|  sw|
    |
57: | 4005f8|  addul  lw|  lw|  sw|  sw|
    |
58: | 4005fc|  sw|  addul  lw|  lw|  sw|
    |
59: | 400600|  slll  sw|  addul  lw|  lw|
    |
60: | 400604|  slll  sw|  nop|  addul  lw|
    |
61: | 400608|  lw|  slll  sw|  nop|  addul
    |
62: | 40060c|  addul  lw|  slll  sw|  nop|
    |
63: | 400610|  lw|  addul  lw|  slll  sw|
    |
64: | 400614|  addiul  lw|  addul  lw|  slll
    |
65: | 400618|  lw|  addiul  lw|  addul  lw|
    |
66: | 40061c|  addul  lw|  addiul  lw|  addul
    |
67: | 400620|  sw|  addul  lw|  addiul  lw|
    |
68: | 400624|  sw|  addul  nop|  lw|  addiul
    |
69: | 400628|  sw|  sw|  addul  nop|  lw|
    |
70: | 40062c|  lw|  sw|  sw|  addul  nop|
    |
71: | 400630|  lw|  sw|  nop|  sw|  addul
    |
72: | 400634|  addul  lw|  sw|  nop|  sw|
```

```

73:| 40062c| bnel addul lwl swl nopl
74:| 400630| addul bnel addul lwl swl
75:| 400630| addul bnel nopl addul lwl
76:| 40063c| lwl addul bnel nopl addul
77:| 400640| addiul lwl addul bnel nopl
78:| 400644| addul addiul lwl addul bnel
79:| 400648| jalr| addul addiul lwl addul
80:| 40064c| addiul jalr| addul addiul lwl
81:| 400e20| sltil addiul jalr| addul addiul
82:| 400e24| bnel sltil addiul jalr| addul
83:| 400e24| bnel sltil nopl addiul jalr|
84:| 400e24| bnel sltil nopl nopl addiul

```

フォワーディング有りの場合

* Pipeline Configuration *

Forwarding: Yes

DataCache Disabled

#####

cycle count: 882

inst count: 786

IPC: 0.891156

simulation time: 0.001

パイプのログをループ本体部分の1ループ分の対応箇所

beql lwl slll bnel nopl

344: | 400830| lwl beql lwl slll bnel

345: | 400834| jalrl lwl beql lwl slll

346: | 400838| nopl jalrl lwl beql lwl

347: | 400838| nopl jalrl nopl lwl beql

348: | 400838| nopl jalrl nopl nopl lwl

349: | 400cc0| lui! nopl jalrl nopl nopl

350: | 400cc4| addiul lui! nopl jalrl nopl

351: | 400cc8| addul addiul lui! nopl jalrl

352: | 400ccc| jrl addul addiul lui! nopl

353: | 400cd0| lwl jrl addul addiul lui!

354: | 40083c| lwl lwl jrl addul addiul

355: | 400840| swl lwl lwl jrl addul

356: | 400844| lwl swl lwl lwl jrl

357: | 400848| beql lwl swl lwl lwl

358: | 40084c| lwl beql lwl swl lwl

359: | 40084c| lwl beql nopl lwl swl

360: | 40084c| lwl beql nopl nopl lwl

361: | 400860| lwl lwl beql nopl nopl

362: | 400864| addul lwl lwl beql nopl

363: | 400868| lwl addul lwl lwl beql

364: | 40086c| addul lwl addul lwl lwl

```

365: | 400870| jalr| addu| lw| addu| lw|
366: | 400874| addu| jalr| addu| lw| addu|
367: | 400874| addu| jalr| nop| addu| lw|
368: | 4002f0| lui| addu| jalr| nop| addu|
369: | 4002f4| addiu| lui| addu| jalr| nop|
370: | 4002f8| addu| addiu| lui| addu| jalr|
371: | 4002fc| addiu| addu| addiu| lui| addu|
372: | 400300| sw| addiu| addu| addiu| lui|
373: | 400304| addu| sw| addiu| addu| addiu|
374: | 400308| sw| addu| sw| addiu| addu|
375: | 40030c| beq| sw| addu| sw| addiu|
376: | 400310| nop| beq| sw| addu| sw|
377: | 400378| lw| nop| beq| sw| addu|
378: | 40037c| nop| lw| nop| beq| sw|
379: | 400380| slti| nop| lw| nop| beq|
380: | 400384| bne| slti| nop| lw| nop|
381: | 400388| nop| bne| slti| nop| lw|
382: | 400388| nop| bne| nop| slti| nop|
383: | 400314| lw| nop| bne| nop| slti|

```

フォーワーディングがあると、演算結果をすぐに次の命令に使用することが出来るのでnop命令を挟む必要がなくなり、ループのために必要なコード数が減っていることがわ

かる。具体的にはbne命令を実行する際にフォーワーディングがない場合はnopを挟んだ後に実行していますが、ある場合はnopを挟まずに分岐条件を実行していることがわかります。

変更前のファイル

```
.file 1 "test07.c"
.section .mdebug.abi32
.previous
.abicalls
.text
.align 2
.globl main
.ent main
.type main, @function
main:
    .frame $fp,24,$31          # vars= 8, regs= 1/0, args= 0, gp= 8
    .mask 0x40000000,-8
    .fmask 0x00000000,0
    .set noreorder
    .cpload$25
    .set nomacro

    addiu $sp,$sp,-24
    sw $fp,16($sp)
    move $fp,$sp
    sw $0,8($fp)
    b $L2
    nop

$L3:
    lw $5,8($fp)
    lw $2,8($fp)
    lw $3,%got(x)($28)
    sll $2,$2,2
    addu $2,$2,$3
    lw $3,0($2)
    nop
    move $2,$3
    sll $2,$2,2
    addu $4,$2,$3
    lw $2,8($fp)
    lw $3,%got(y)($28)
```

```

    sll    $2,$2,2
    addu   $2,$2,$3
    lw     $2,0($2)
    nop
    addu   $4,$4,$2
    lw     $3,%got(y)($28)
    sll    $2,$5,2
    addu   $2,$2,$3
    sw     $4,0($2)
    lw     $2,8($fp)
    nop
    addiu  $2,$2,1
    sw     $2,8($fp)
$L2:
    lw     $2,8($fp)
    nop
    slt    $2,$2,10
    bne    $2,$0,$L3
    nop

    move   $2,$0
    move   $sp,$fp
    lw     $fp,16($sp)
    addiu  $sp,$sp,24
    j      $31
    nop

.set      macro
.set      reorder
.end      main

.comm x,40,4

.comm y,40,4
.ident "GCC: (GNU) 4.2.4"
変更後のファイル
.file     1 "test07.c"
.section .mdebug.abi32
.previous
.abicalls
.text
.align 2
.globl main
.ent     main
.type    main, @function
main:

```

```

.frame $fp,24,$31          # vars= 8, regs= 1/0, args= 0, gp= 8
.mask 0x40000000,-8
.fmask 0x00000000,0
.set    noreorder
.cpload$25
.set    nomacro

```

```

addiu   $sp,$sp,-24
sw      $fp,16($sp)
move    $fp,$sp
sw      $0,8($fp)
b       $L2
nop

```

\$L3:

```

lw      $5,8($fp)
lw      $2,8($fp)
lw      $3,%got(x)($28)
sll     $2,$2,2
addu    $2,$2,$3
lw      $3,0($2)
move    $2,$3
sll     $2,$2,2
addu    $4,$2,$3
lw      $2,8($fp)
lw      $3,%got(y)($28)
sll     $2,$2,2
addu    $2,$2,$3
lw      $2,0($2)
addu    $4,$4,$2
lw      $3,%got(y)($28)
sll     $2,$5,2
addu    $2,$2,$3
sw      $4,0($2)
lw      $2,8($fp)
addiu   $2,$2,1
sw      $2,8($fp)

```

\$L2:

```

lw      $2,8($fp)
slt     $2,$2,10
bne     $2,$0,$L3
nop

```

```

move    $2,$0
move    $sp,$fp
lw      $fp,16($sp)

```

```

    addiu    $sp,$sp,24
    j        $31
    nop

    .set     macro
    .set     reorder
    .end     main

    .comm x,40,4

    .comm y,40,4
    .ident   "GCC: (GNU) 4.2.4"

```

変更点 : lw命令の後のすべてのnop命令を削除した

フォーワーディングなしの場合

```

* Pipeline Configuration *
Forwarding: No
DataCache Disabled

```

```

#####
## cycle count: 1366
## inst count: 745
## IPC: 0.545388
## simulation time: 0.001

```

フォーワーディング有りの場合

```

* Pipeline Configuration *
Forwarding: Yes
DataCache Disabled

```

```

#####
## cycle count: 882
## inst count: 745
## IPC: 0.844671
## simulation time: 0.001

```

どちらのpipe.logファイルに変更はなかったが,inst countとIPCは共に小さくなった
 原因はcycle countはnop命令がなくなっても変わらず、nop命令がないことによって1サイ
 クル辺りの処理命令数(IPC)が減る。また処理命令数が減ることで命令カウンターの値

(inst count)も小さくなる。従ってcycle countの値は $\text{inst count}/\text{IPC} \approx 882$ と変化はない。nop命令は組み込むことで見た目上処理命令能力を上げてゐる事が出来るとわかる。

2.参考文献

- ・ 「Comparing Computer Performance Using Execution Time」 <http://meseec.ce.rit.edu/eccc550-winter2011/550-12-6-2011.pdf><2019/11/20>