EE 502

Computer Architecture

Lab Assignment #1

Zexi Liu

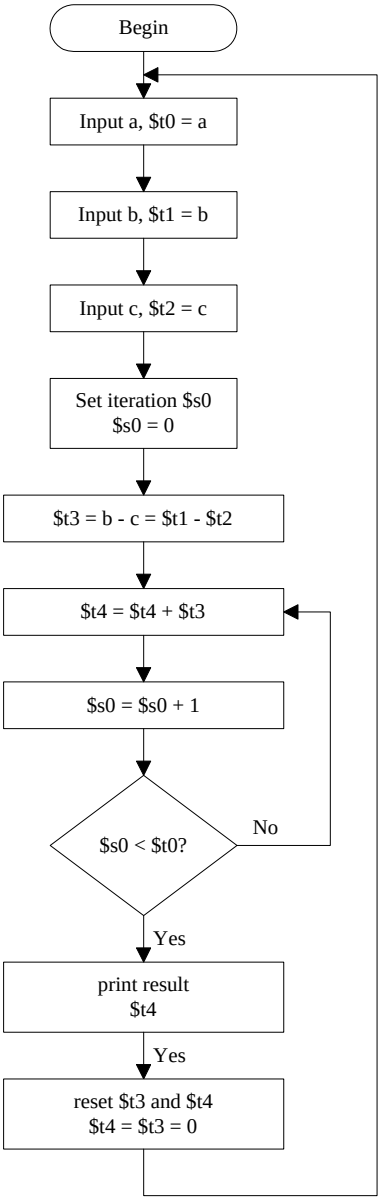
Feb 10th, 2007

Electrical and Computer Engineering

Temple University

1. **(20 points) In SPIM, write a MIPS program that calculates d=a*b-a*c without using mult instruction**

1.1 Flowchart



1.2 Source Code

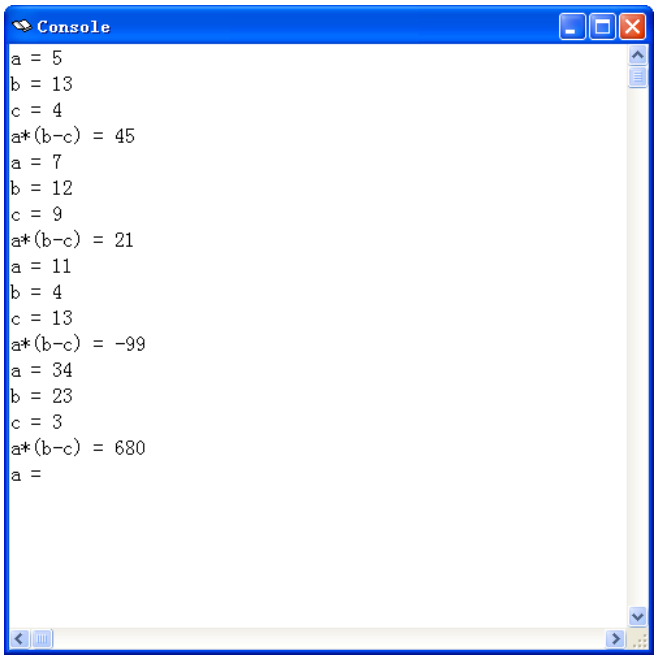
```
# ---------------------------------------------------------------------
# A simple MIPS program calculates d=a*b-a*c=a*(b-c).
# ---------------------------------------------------------------------
.data
ans:    .asciiz "a*(b-c) = "        # strings to be printed
_a:     .asciiz "a = "              # strings to be printed
_b:     .asciiz "b = "              # strings to be printed
_c:     .asciiz "c = "              # strings to be printed
newln:  .asciiz "\n"                # carrier return
        .text

cr:     # print newline
        li      $v0, 4              # system call for print_str
        la      $a0, newln          # address of the string to be printed
        syscall                     # print_str
        jr      $ra                 # return

main:                               # main program starts with label "main"
        li      $v0, 4              # system call for print_str
        la      $a0, _a             # address of the string to be printed
        syscall                     # system call for print a string
        # input a
        li      $v0, 5              # system call for read_int
        syscall                     # system call for read an integer
        add     $t0, $v0, $zero     # $t0 = $v0
        # input b
        li      $v0, 4              # system call for print_str
        la      $a0, _b             # address of the string to be printed
        syscall                     # system call for print a string
        li      $v0, 5              # system call for read_int
        syscall                     # system call for read an integer
        add     $t1, $v0, $zero     # $t1 = $v0
        # input c
        li      $v0, 4              # system call for print_str
        la      $a0, _c             # address of the string to be printed
        syscall                     # system call for print a string
        li      $v0, 5              # system call for read_int
        syscall                     # system call for read an integer
        add     $t2, $v0, $zero     # $t2 = $v0
        # calculation
        addi    $s0, $zero, 0       # i = 0
        sub     $t3, $t1, $t2       # $t3 = $t1 - $t2
```

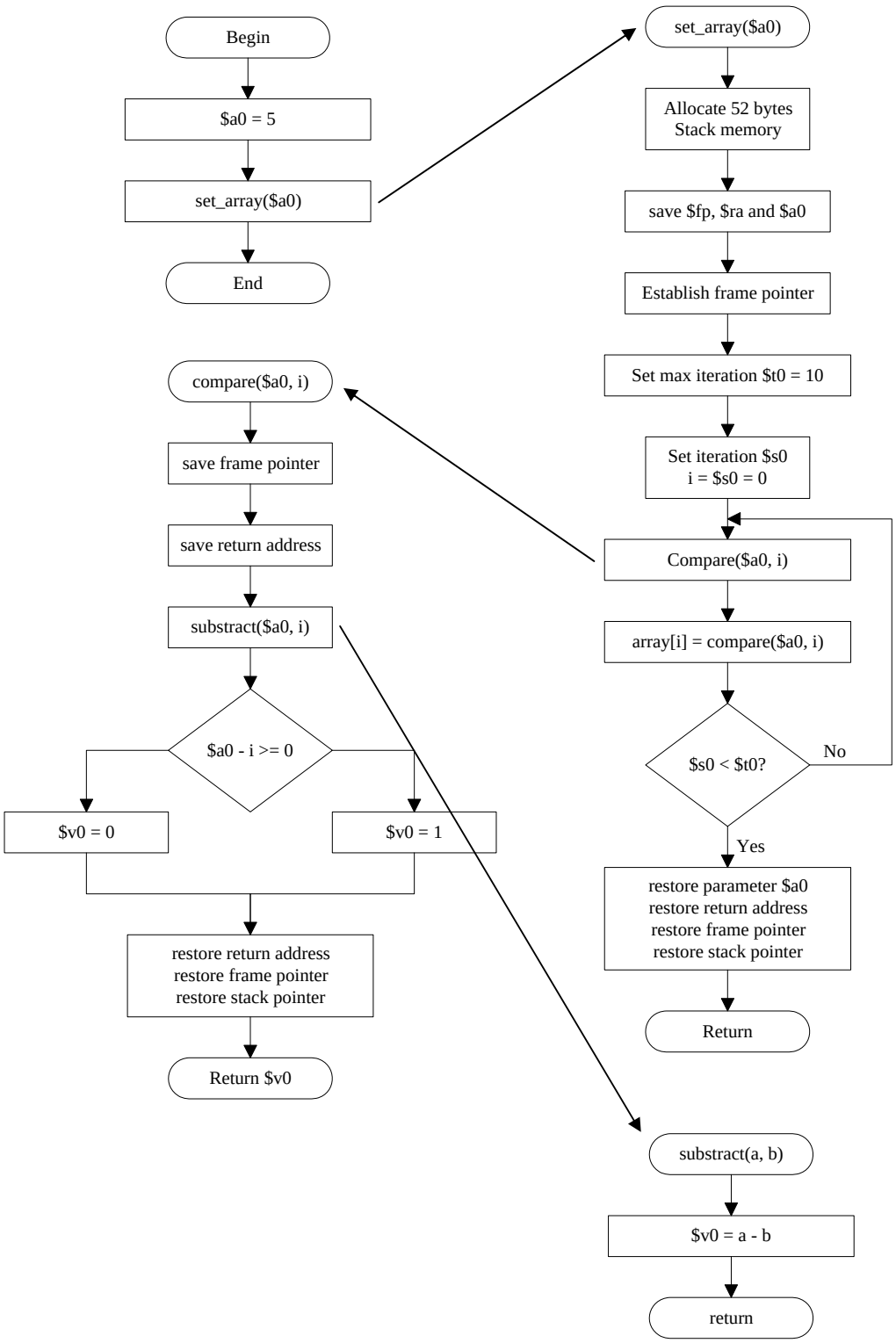
```
loop:   add     $t4, $t4, $t3       # $t4 = $t4 + $t3
        addi    $s0, $s0, 1         # $s0 ++
        bne     $s0, $t0, loop      # loop if i < $t0
        # print result
        li      $v0, 4              # system call for print_str
        la      $a0, ans            # address of the string to be printed
        syscall                     # system call for print a string
        li      $v0, 1              # system call for print_int
        add     $a0, $t4, $zero     # integer must be in $a0 for print_int
        syscall                     # print_int
        jal     cr                  # print new line
        add     $t3, $zero, $zero   # reset $t3 for next calculation
        add     $t4, $zero, $zero   # reset $t4 for next calculation
        j       main                # next calculation
        .end
```

1.3 Results



2. (40 points) Using SPIM to implement MIPS assembly program for set_arrary() of Chap 2. Exercise 2.15.

2.1 Flowchart



2.2 Source Code

```
# ---------------------------------------------------------------------
# A MIPS program process an array.
# ---------------------------------------------------------------------
                .data
                .text

set_array:      addi    $sp, $sp, -52       # move stack pointer
                sw      $fp, 48($sp)        # save frame pointer
                sw      $ra, 44($sp)        # save return address
                sw      $a0, 40($sp)        # save parameter num
                addi    $fp, $sp, 48        # establish frame pointer

                add     $s0, $zero, $zero   # i = 0
                addi    $t0, $zero, 10      # max iterations is 10
loop:           sll     $t1, $s0, 2         # $t1 = i * 4
                add     $t2, $sp, $t1       # $t2 = address of array[i]
                add     $a0, $a0, $zero     # pass num as parameter
                add     $a1, $s0, $zero     # pass i as parameter
                jal     compare             # call compare(num, i)
                sw      $v0, 0($t2)         # array[i] = compare(num, i);
                addi    $s0, $s0, 1
                bne     $s0, $t0, loop      # loop if i<10

                lw      $a0, 40($sp)        # restore parameter (num)
                lw      $ra, 44($sp)        # restore return address
                lw      $fp, 48($sp)        # restore frame pointer
                addi    $sp, $sp, 52        # restore stack pointer
                jr      $ra                 # return

compare:        addi    $sp, $sp, -8        # move stack pointer
                sw      $fp, 4($sp)         # save frame pointer
                sw      $ra, 0($sp)         # it save return address
                addi    $fp, $sp, 4         # establish frame pointer

                jal     subtract            # can jump directly to subtract
                slt     $v0, $v0, $zero     # if sub(a,b) >= 0, return 1
                slti    $v0, $v0, 1

                lw      $ra, 0($sp)         # restore return address
                lw      $fp, 4($sp)         # restore frame pointer
                addi    $sp, $sp, 8         # restore stack pointer
                jr      $ra                 # return
```

```
subtract:   sub     $v0, $a0, $a1     # return a-b
            jr      $ra               # return

main:       # main
            addi    $a0, $a0, 5       # $a0 = 5
            jal     set_array

            .end
```

2.3 Result

2.3.1 Before calling set_array()

```
 PC      = 004000ac   EPC     = 004000ac   Cause   = 00000024   BadVAddr= 00000000
 Status  = 3000ff10   HI      = 00000000   LO      = 00000000
                                General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000005  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 7fffef3c  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp) = 7fffef38
R6  (a2) = 7fffef40  R14 (t6) = 00000000  R22 (s6) = 00000000  R30 (s8) = 00000000
R7  (a3) = 00000000  R15 (t7) = 00000000  R23 (s7) = 00000000  R31 (ra) = 00400018


        DATA
[0x10000000]...[0x10010000]     0x00000000
[0x10010000]                    0x0000000a  0x00000003  0x00000004  0x00000000
[0x10010010]...[0x10040000]     0x00000000

        STACK
[0x7fffef38]                    0x00000000  0x00000000
[0x7fffef40]                    0x7fffefc9  0x7fffef97  0x7fffef60  0x7fffef17
[0x7fffef50]                    0x7fffeee6  0x7fffeec9  0x7fffeea5  0x7fffee91
[0x7fffef60]                    0x7fffee84  0x7fffee5f  0x7fffedad  0x7fffed74
[0x7fffef70]                    0x7fffed08  0x7fffecce  0x7fffecb0  0x7fffec6f
[0x7fffef80]                    0x7fffec30  0x7fffec19  0x7fffec0b  0x7fffebcb
```

2.3.2 During set_array()

```
 PC      = 00400060   EPC     = 00400060   Cause   = 00000024   BadVAddr= 00000000
 Status  = 3000ff10   HI      = 00000000   LO      = 00000000
                                General Registers
R0  (r0) = 00000000  R8  (t0) = 0000000a  R16 (s0) = 0000000a  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000024  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 7fffef28  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000005  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 00000009  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp) = 7fffef04
R6  (a2) = 7fffef40  R14 (t6) = 00000000  R22 (s6) = 00000000  R30 (s8) = 7fffef34
R7  (a3) = 00000000  R15 (t7) = 00000000  R23 (s7) = 00000000  R31 (ra) = 00400054


        DATA
[0x10000000]...[0x10010000]     0x00000000
[0x10010000]                    0x0000000a  0x00000003  0x00000004  0x00000000
[0x10010010]...[0x10040000]     0x00000000

        STACK
[0x7fffef04]                    0x00000001  0x00000001  0x00000001
[0x7fffef10]                    0x00000001  0x00000001  0x00000001  0x00000000
[0x7fffef20]                    0x00000000  0x00000000  0x00000000  0x00000005
[0x7fffef30]                    0x004000b0  0x00000000  0x00000000  0x00000000
[0x7fffef40]                    0x7fffefc9  0x7fffef97  0x7fffef60  0x7fffef17
[0x7fffef50]                    0x7fffeee6  0x7fffeec9  0x7fffeea5  0x7fffee91
```

2.3.3 During compare()

```
 PC      = 0040009c   EPC     = 0040009c   Cause   = 00000024   BadVAddr= 00000000
 Status  = 3000ff10   HI      = 00000000   LO      = 00000000
                                General Registers
R0  (r0) = 00000000  R8  (t0) = 0000000a  R16 (s0) = 00000002  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000008  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000001  R10 (t2) = 7fffef0c  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000005  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 00000002  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp) = 7fffef04
R6  (a2) = 7fffef40  R14 (t6) = 00000000  R22 (s6) = 00000000  R30 (s8) = 7fffef34
R7  (a3) = 00000000  R15 (t7) = 00000000  R23 (s7) = 00000000  R31 (ra) = 00400054


        DATA
[0x10000000]...[0x10010000]     0x00000000
[0x10010000]                    0x0000000a  0x00000003  0x00000004  0x00000000
[0x10010010]...[0x10040000]     0x00000000

        STACK
[0x7fffef04]                    0x00000001  0x00000001  0x6e6f6d6d
[0x7fffef10]                    0x6c694620  0x43007365  0x5353414c  0x48544150
[0x7fffef20]                    0x5c3a663d  0x676f7250  0x206d6172  0x00000005
[0x7fffef30]                    0x004000b0  0x00000000  0x00000000  0x00000000
[0x7fffef40]                    0x7fffefc9  0x7fffef97  0x7fffef60  0x7fffef17
[0x7fffef50]                    0x7fffeee6  0x7fffeec9  0x7fffeea5  0x7fffee91
```

2.3.4 During sub()

```
 PC      = 004000a4   EPC     = 004000a4   Cause   = 00000024   BadVAddr= 00000000
 Status  = 3000ff10   HI      = 00000000   LO      = 00000000
                                General Registers
R0  (r0) = 00000000  R8  (t0) = 0000000a  R16 (s0) = 00000002  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000008  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000003  R10 (t2) = 7fffef0c  R18 (s2) = 00000000  R26 (k0) = 00000000
R3  (v1) = 00000000  R11 (t3) = 00000000  R19 (s3) = 00000000  R27 (k1) = 00000000
R4  (a0) = 00000005  R12 (t4) = 00000000  R20 (s4) = 00000000  R28 (gp) = 10008000
R5  (a1) = 00000002  R13 (t5) = 00000000  R21 (s5) = 00000000  R29 (sp) = 7fffeefc
R6  (a2) = 7fffef40  R14 (t6) = 00000000  R22 (s6) = 00000000  R30 (s8) = 7fffef00
R7  (a3) = 00000000  R15 (t7) = 00000000  R23 (s7) = 00000000  R31 (ra) = 00400088


        DATA
[0x10000000]...[0x10010000]     0x00000000
[0x10010000]                    0x0000000a  0x00000003  0x00000004  0x00000000
[0x10010010]...[0x10040000]     0x00000000

        STACK
[0x7fffeefc]                    0x00400054
[0x7fffef00]                    0x7fffef34  0x00000001  0x00000001  0x6e6f6d6d
[0x7fffef10]                    0x6c694620  0x43007365  0x5353414c  0x48544150
[0x7fffef20]                    0x5c3a663d  0x676f7250  0x206d6172  0x00000005
[0x7fffef30]                    0x004000b0  0x00000000  0x00000000  0x00000000
[0x7fffef40]                    0x7fffefc9  0x7fffef97  0x7fffef60  0x7fffef17
```

3. **(90 points) Using sim-profile of SimpleScalar's SimpleSim-ARM and MiBench to generate instruction profiles of all 3 SMALL benchmarks of CRC32 and jpeg.**

3.1 CRC

Command: sim-profile -all -redir:sim simout.doc crc.arm small.pcm > output_small.txt

Content of output_small.txt: 6DA5B639 1368864 small.pcm

Class of Instructions	Number	Percentage
load	8262237	13.37 %
store	12358239	20 %
unconditional branch	4112300	6.65 %
conditional branch	5497758	8.9 %
integer computation	31570321	51.08 %
floating point computation	0	0 %
trap	1359	0 %
Total Instructions	61802214	

Individual Instruction	Number	Percentage
b%c	5499212	8.9 %
bl%c	4110846	6.65 %
swi%c	1359	0 %
and%c	1	0 %
eor%c	2737736	4.43 %
sub%c	256	0 %
sub%cs	92	0 %
rsb%c	1468	0 %
add%c	6809	0.01 %
tst%cs	24	0 %
cmp%c	1375239	2.23 %
orr%c	716	0 %
orr%cs	3	0 %
mov%c	12353421	19.99 %
mov%cs	71	0 %
bic%cs	6	0 %
mvn%c	6	0 %
and%c	1368896	2.21 %
and%cs	27	0 %
eor%c	1	0 %
sub%c	5482477	8.87 %

sub%cs	1368965	2.22 %
rsb%c	18	0 %
add%c	1369161	2.22 %
tst%cs	5653	0.01 %
cmp%c	2747428	4.45 %
cmn%c	1374256	2.22 %
orr%c	38	0 %
mov%c	1372103	2.22 %
bic%c	4080	0.01 %
bic%cs	3	0 %
mvn%c	1365	0 %
ldr%c	1	0 %
str%c	84	0 %
ldr%c	31	0 %
str%cb	1369155	2.22 %
ldr%cb	105	0 %
str%c	140	0 %
ldr%c	30	0 %
ldr%cb	20	0 %
str%cb	2756696	4.46 %
ldr%cb	5514072	8.92 %
str%c	525	0 %
ldr%c	11	0 %
ldr%c	7905	0.01 %
str%cb	86	0 %
ldr%cb	7	0 %
ldr%cb	1370221	2.22 %
str%c	2	0 %
ldr%c	19	0 %
str%cb	11	0 %
ldr%cb	14	0 %
ldm%c%a	2	0 %
stm%c%a	2	0 %
ldm%c%a	4	0 %
stm%c%a	4116288	6.66 %
ldm%c%a	4113503	6.66 %
ldm%c%a	1	0 %
stm%c%a	1371541	2.22 %
stm%c%a	1	0 %

ldm%c%a	1	0 %
wfs%c	5499212	8.9 %
rfs%c	4110846	6.65 %
Total Instruction	61802214	

3.2 cjpeg (encode)

Command: sim-profile -all -redir:sim simout.doc cjpeg.arm -dct int -progressive -opt -outfile output_small_encode.jpeg input_small.ppm

Output file: output_small_encode.jpeg

Class of Instructions	Number	Percentage
load	7228874	25.69 %
store	2324035	8.26 %
unconditional branch	366700	1.3 %
conditional branch	2707618	9.62 %
integer computation	15511808	55.13 %
Floating point computation	0	0 %
trap	227	0 %
Total Instructions	28139262	

Individual Instruction	Number	Percentage
b%c	3012748	10.71 %
bl%c	61570	0.22 %
swi%c	227	0 %
and%c	20150	0.07 %
eor%c	13184	0.05 %
sub%c	112064	0.4 %

sub%cs	2163	0.01 %
rsb%c	819015	2.91 %
add%c	2719922	9.67 %
tst%cs	12	0 %
cmp%c	1462083	5.2 %
orr%c	525714	1.87 %
orr%cs	193541	0.69 %
mov%c	3195510	11.36 %
mov%cs	50641	0.18 %
bic%c	1	0 %
bic%cs	10	0 %
mvn%c	28889	0.1 %
and%c	42904	0.15 %
and%cs	150	0 %
eor%c	32768	0.12 %
sub%c	667129	2.37 %
sub%cs	52615	0.19 %
rsb%c	488180	1.73 %
rsb%cs	1	0 %
add%c	2562409	9.11 %
tst%cs	2196	0.01 %
cmp%c	2017322	7.17 %
cmn%c	892	0 %
orr%c	91	0 %
mov%c	485805	1.73 %
bic%c	1897	0.01 %
mvn%c	620	0 %
mul%c	13669	0.05 %
mul%cs	259	0 %
ldr%c	4	0 %
str%c	98497	0.35 %
ldr%c	855	0 %
str%cb	43517	0.15 %
ldr%cb	75087	0.27 %
str%c	718474	2.55 %
ldr%c	2630206	9.35 %
str%c	6	0 %
ldr%cb	42	0 %
str%cb	5	0 %

str%c	438393	1.56 %
ldr%c	1319609	4.69 %
ldr%c	644	0 %
str%cb	98589	0.35 %
ldr%cb	892635	3.17 %
ldr%cb	85	0 %
str%c	363906	1.29 %
ldr%c	1763005	6.27 %
str%cb	391553	1.39 %
ldr%cb	546702	1.94 %
ldm%c%a	5580	0.02 %
stm%c%a	32	0 %
ldm%c%a	12774	0.05 %
stm%c%a	3154	0.01 %
ldm%c%a	15	0 %
ldm%c%a	85188	0.3 %
stm%c%a	63775	0.23 %
stm%c%a	68	0 %
ldm%c%a	509	0 %
wfs%c	1	0 %
rfs%c	1	0 %
Total Instructions	28139262	

3.3 djpeg (decode)

Command: sim-profile -all -redir:sim simout.doc djpeg.arm -dct int -ppm -outfile output_small_encode.ppm input_small.jpg

Output file: output_small_encode.ppm

Class of Instructions	Number	Percentage
load	2060547	30.7 %
store	801660	11.94 %
unconditional branch	39821	0.59 %
conditional branch	334678	4.99 %
integer computation	3474449	51.77 %
Floating point computation	0	0 %
trap	218	0 %
Total Instructions	6711373	

Individual Instruction	Number	Percentage
b%c	367991	5.48
bl%c	6508	0.1
swi%c	218	0
and%c	13010	0.19
eor%c	63	0
sub%c	1736	0.03
sub%cs	90	0
rsb%c	322401	4.8
add%c	1120969	16.7
tst%cs	32	0
cmp%c	101601	1.51
orr%c	294731	4.39
orr%cs	24773	0.37
mov%c	692030	10.31
mov%cs	491	0.01
bic%cs	10	0
mvn%c	7	0
and%c	12510	0.19
and%cs	8326	0.12
eor%c	16	0
sub%c	89761	1.34
sub%cs	81952	1.22
rsb%c	11177	0.17
add%c	507026	7.55
tst%cs	1689	0.03
cmp%c	129946	1.94
cmn%c	820	0.01
orr%c	270	0
mov%c	26826	0.4
bic%c	1281	0.02
bic%cs	6	0
mvn%c	815	0.01
mul%c	29821	0.44
mul%cs	258	0
mla%c	3	0
str%c	1	0
ldr%c	10419	0.16
str%cb	132655	1.98

ldr%cb	140183	2.09
str%c	119691	1.78
ldr%c	582264	8.68
str%c	2	0
str%cb	4	0
ldr%cb	104	0
str%cb	14	0
str%c	164852	2.46
ldr%c	300378	4.48
ldr%c	769	0.01
str%cb	300498	4.48
ldr%cb	230331	3.43
ldr%cb	85	0
str%c	5304	0.08
ldr%c	325752	4.85
str%cb	14817	0.22
ldr%cb	470262	7.01
ldm%c%a	847	0.01
stm%c%a	723	0.01
ldm%c%a	15996	0.24
stm%c%a	25583	0.38
ldm%c%a	7	0
ldm%c%a	10682	0.16
stm%c%a	9727	0.14
stm%c%a	257	0
wfs%c	1	0
rfs%c	1	0
Total Instructions	6711373	