Hitachi America, Ltd.

Application Note

GNU H8/300 Software

Jennifer Ediyanto

GNU Program Optimizations

INTRODUCTION

Optimizations are techniques performed by the compiler to improve a program in reducing code size, increasing execution speed, compilation time, and the performance of the generated code. The extent of improvement depends on the content of an individual program, its coding style, and the compiler's ability to recognize and optimize certain constructs.

Benefits of optimization for Embedded Systems:

- Faster code gives better real-time response.
- More compact code makes better use of limited resources.
- Reduce tendency to use assembly language (thus enhancing maintainability).

This paper provides some explanations on the GNU C compiler's optimizations. The GNU optimizations are divided into two groups:

- Optimize switches: `-O` and `-O1`. The compiler tries to reduce code size and execution time.
- Optimize even more switches: `-O2, -O3, -O4`, and `-O5`. The compiler increases both compilation time and the performance of the generated code.

OPTIMIZATIONS OPTIONS

All `-O` switches are controlled by the `-fflag` options. Most flags have both positive and negative forms; the negative form of `-fdefer-pop` will be `-fno-defer-pop`. When we turn on the `-O` switch, most of the `-flag` options are enabled.

The following are the options that control optimizations:

1. **-ffloat-store**, **-fno-float-store** (default).

The -ffloat-store option stores floating point variables in registers. Using floating registers will result in larger code size because floating registers keep more precision. Thus, it is better to disable this option.

Example:

```
  float k=0.123456789;
```

The assembly code with -ffloat-store option:

```
LC0:
   .float 1.23456791043281555176e-1
_test:
   mov.w  #6,r3
   sub.w  r3,sp
   push   r5
   push   r4
   mov.w  r0,@(4,r7)
   mov.w  @.LC0,r4
   mov.w  @.LC0+2,r5
   mov.w  r4,@(6,r7)
   mov.w  r5,@(8,r7)
   mov.b  @r0,r2l
```

The assembly code with -fno-float-store option:

```
LC0:
   .float 1.23456791043281555176e-1
_test:
   push r4
   mov.w  r0,r4
   mov.w  @.LC0,r0
   mov.w  @.LC0+2,r1
```

Please see *Listing 1* for the C sample code and the compiler generated assembly code with -ffloat-store and -fno-float-store comparison.

2. **-fdefer-pop** (default), **-fno-defer-pop**.

The -fdefer-pop option lets arguments accumulate on the stack for several function calls and pops them all at once.

Example:

```
   foo(1,2,3,4,5);
   foo(1,2,3,4,5);
```

The assembly code with -fdefer-pop option:

```
_test:
   push    r5
   push    r4
   mov.w   #5,r5
   push.w  r5
   mov.w   #4,r4
   push.w  r4
   mov.w   #3,r2
   mov.w   #2,r1
   mov.w   #1,r0
   jsr     @_foo
```

The assembly code with -fno-defer-pop option:

```
_test:
   push    r5
   push    r4
   mov.w   #5,r5
   push.w  r5
   mov.w   #4,r4
   push.w  r4
   mov.w   #3,r2
   mov.w   #2,r1
   mov.w   #1,r0
   jsr     @_foo
   adds    #2,r7 ;accumulate arguments
   adds    #2,r7
```

Please see *Listing 2* for the C sample code and the compiler generated assembly code with -fdefer-pop and -fno-defer-pop comparison.

3. **-fforce-mem**, **-fno-force-mem** (default).

The -force-mem option forces memory operands to be copied into registers before doing arithmetic on them. This may produce better code by making all memory references potential common subexpressions.

Example:

```
   k += *p;
```

The assembly code with -fforce-mem option:

```
   mov.b   @r0,r2l
   mov.b   #0,r2h
   mov.w   @_k,r3
   add.w   r2,r3
   mov.w   r3,r2  ;copy to register
   mov.w   r2,@_k
```

The assembly code with -fno-force-mem option:

```
   mov.b   @r0,r2l
   mov.b   #0,r2h
   mov.w   @_k,r3
   add.w   r2,r3
   mov.w   r2,@_k
```

Please see *Listing 3* for the C sample code and the compiler generated assembly code with -fforce-mem and -fno-force-mem comparison.

4. **-fforce-addr**, **-fno-force-addr** (default).

The -fforce-mem option forces memory address constants to be copied into registers before doing arithmetic on them. This may produce better code by making all memory references potential common subexpressions.

Example:

```
   int j;
   j = 1;
```

The assembly code with -fforce-addr option:

```
   push    r4
   mov.w   #_j,r3   ;copy to register
   mov.w   #1,r4
   mov.w   r4,@r3
```

The assembly code with -fno-force-addr option:

```
   mov.w    #1,r3
   mov.w    r3,@_j
```

Please see *Listing 4* for the C sample code and the compiler generated assembly code with -fforce-addr and -fno-force-addr comparison.

5. **-fomit-frame-pointer**,
   -fno-omit-frame-pointer (default).

The -fomit-frame-pointer prevents keeping frame pointer in a register for functions that do not need one. This avoids the instructions to save, set up, and restore frame pointers. It also makes an extra register available in many functions.

Example:

```
   test(char *p)
   {
   }
```

The assembly code with -fomit-frame-pointer option:

```
_test:
   sub.w   r2,r2
   mov.b   @r0,r0l
```

The assembly code with -fno-omit-frame-pointer option:

```
_test:
   push    r6     ;keep frame pointer
   mov.w   r7,r6  ;in a register
   sub.w   r2,r2
   mov.b   @r0,r0l
```

Please see *Listing 5* for the C sample code and the compiler generated assembly code with -fomit-frame-pointer and -fno-omit-frame-pointer comparison.

6. **-finline** (default), **-fno-inline**.

The -finline makes compiler to expand functions inline. Without optimization, no functions can be expanded inline.

Example:

```
   inline
   foo(int p)
   {
    return p * 5;
   }

   test(char *p)
   {
    int k = 100;
    k = foo(k);
    return k;
   }
```

The assembly code with -finline option:

```
_foo:
   mov.w   r0,r2
   add.w   r0,r0
   add.w   r0,r0
   add.w   r2,r0
   rts
_test:
   mov.w   #500,r0  ;execute directly
   rts
   .end
```

The assembly code with -fno-inline option:

```
_foo:
   mov.w   r0,r2
   add.w   r0,r0
   add.w   r0,r0
   add.w   r2,r0
   rts
_test:
   mov.w   #100,r0
   jsr     @_foo    ;call function foo
   rts
   .end
```

Please see *Listing 6* for the C sample code and the compiler generated assembly code with -finline and -fno-inline comparison.

7. **-finline-functions**, **-fno-inline-functions** (default).

The -finline-functions option integrates all simple functions into their callers. The compiler decides which functions are simple enough to be worth integrating in this way. The compiler will not produce assembly code for inline function that is declared as static.

Example:

```
   int j;
   int k;
   foobar(int *a, int b, int c)
   {
     j = 1;
     a[b] = c;
     j++;
   }
   test()
   {
     j = 100;
     foobar(&k,&j-&k,12345);
   }
```

The assembly code with -finline-functions option:

```
_test:
   mov.w   #_j,r2
   mov.w   #_k,r3
   sub.w   r3,r2
   mov.w   #1,r3
   mov.w   r3,@_j
   and     #254,r2l
   mov.w   #12345,r3
   mov.w   r3,@(_k,r2)
   mov.w   #2,r3
   mov.w   r3,@_j
   rts
```

The assembly code with -fno-inline-functions option:

```
_test:
   mov.w   #100,r3
   mov.w   r3,@_j
   mov.w   #_k,r0
   mov.w   #_j,r1
   sub.w   r0,r1
   shar    r1h
   rotxr   r1l
   mov.w   #12345,r2
   jsr     @_foobar ;callfunctionfoobar
   rts
```

Please see *Listing 7* for the C sample code and the compiler generated assembly code with -finline-functions and -fno-inline-functions comparison.

8. **-fkeep-inline-functions**,
-fno-keep-inline-functions (default).

The -fkeep-inline-functions option will produce the assembly code for inline function that is declared as static.

Example:

```
   static inline
   foo()
   {
     return 0;
   }
   test(char *p)
   {
     return foo();
   }
```

The assembly code with -fkeep-inline-functions option:

```
_foo:
   sub.w   r0,r0
   rts
_test:
   sub.w   r0,r0
   rts
   end
```

The assembly code with -fno-keep-inline-functions:

```
   ;no assembly code generated for foo
_test:
   sub.w   r0,r0
   rts
   end
```

Please see *Listing 8* for the C sample code and the compiler generated assembly code with -f-keep-inline-functions and -fno-keep-inline-functions comparison.

9. **-fstrength-reduce**, **-fno-strength-reduce** (default).

The -fstength-reduce option performs the optimizations of loop strength reduction and elimination of iteration variables.

Example:

```
   test (short a, int *p, short *s)
   {
     int i;
     for(i=10;i>=0;i--)
     {
       a = (short) bar();
       p[i] = a;
       s[i] = a;
     }
   }
```

The assembly code with -fstrength-reduce option:

```
_test:
   subs    #2,sp
   push    r5
   push    r4
   mov.w   #10,r3
   mov.w   r2,r5
   add.b   #20,r5l
   addx    #0,r5h
   mov.w   r1,r4
   add.b   #20,r4l
   addx    #0,r4h
```

The assembly code with -fno-strength-reduce option:

```
_test:
   subs    #2,sp
   push    r5
   push    r4
   mov.w   r1,@(4,r7)
   mov.w   r2,r5
   mov.w   #10,r4
```

Please see *Listing 9* for the C sample code and the compiler generated assembly code with -fstrength-reduce and -fno-strength-reduce comparison.

10. **-fthread-jumps**, **-fno-thread-jumps** (default).

The -fthread-jumps option performs optimizations where the compiler checks to see if a jump branches to a location where another comparison subsumed by the first is found. If so, the first branch is redirected to either the destination of the second branch or a point immediately following it, depending on whether the condition is known to be true or false.

Example:

```
   test (char *p)
   {
     int i, k, j=2;
     k = *p;
     if (k)
     { j++; }
     if (!k)
     { k--; }
     return k+j;
   }
```

The assembly code with -fthread-jumps option:

```
_test:
   mov.w    #2,r2
   mov.b    @r0,r0l
   mov.b    #0,r0h
   mov.w    r0,r0
   beq      .L4
   mov.w    #3,r2
   bra      .L3
.L4:
   mov.w    #-1,r0
.L3:
   add.w    r2,r0
   rts
```

The assembly code with -fno-thread-jumps option:

```
_test:
   mov.w   #2,r2
   mov.b   @r0,r0l
   mov.b   #0,r0h
   mov.w   r0,r0
   beq     .L2
   mov.w   #3,r2
.L2:
   mov.w   r0,r0
   bne     .L3      ;additional branch
   mov.w   #-1,r0
.L3:
   add.w   r2,r0
   rts
```

Please see *Listing 10* for the C sample code and the compiler generated assembly code with -fthread-jumps and -fno-thread-jumps comparison.

11. **-fcse-follow-jumps**, **-fno-cse-follow-jumps** (default).

The -fcse-follow-jumps option scans through jump instructions when the target of the jump is not reached by any other path. For example, when the compiler encounters an 'if' statement with an 'else' clause, the compiler will follow the jump when the condition tested is false.

Example:

```
   test (char *p)
   {
     int a = 0;
     if (*p == a)
       return 0;
     else
       return 1;
   }
```

The assembly code with -fcse-follow-jumps option:

```
_test:
   mov.b   @r0,r0l
   mov.b   #0,r0h
   mov.w   r0,r0
   beq     .L2
   mov.w   #1,r0
.L2:
   rts
```

The assembly code with -fno-cse-follow-jumps option:

```
_test:
   mov.b   @r0,r2l
   mov.b   #0,r2h
   mov.w   r2,r2
   beq     .L2
   mov.w   #1,r0
   bra     .L4
.L2:
   sub.w   r0,r0
.L4:
   rts
```

Please see *Listing 11* for the C sample code and the compiler generated assembly code with fcse-follow-jumps and fno-cse-follow-jumps comparison.

12. **-fcse-skip-blocks**, **-fno-cse-skip-blocks** (default).

The -fcse-skip-blocks option causes the compiler to follow jumps which conditionally skip over blocks. When the compiler encounter a simple 'if' statement with no 'else' clause, it follows the jump around the body of the 'if'.

Example:

```
   if (124 & *a)
     b =  3;
   if (~111 & *a)
     b =  4;
```

The assembly code with -fcse-skip-blocks option:

```
.L3:
   mov.b   @r3,r2l
   and     #124,r2l
   beq     .L4
   mov.w   #3,r0
.L4:
   mov.w   r4,r2
   and     #144,r2l
   mov.w   r2,r2
   beq     .L5
   mov.w   #4,r0
```

The assembly code with -fno-cse-skip-blocks option:

```
.L3:
   mov.b   @r3,r2
   adds    #1,r2   ;-fcse-skip-blocks
                   ;skip over
   mov.b   @r2,r2l ;these statements
   and     #124,r2l
   beq     .L4
   mov.w   #3,r0
.L4:
   mov.w   r3,r2
   and     #144,r2l
   mov.w   r2,r2
   beq     .L5
   mov.w   #4,r0
```

Please see *Listing 12* for the C sample code and the compiler generated assembly code with -fcse-skip-blocks and -fno-cse-skip-blocks comparison.

13. **-frerun-cse-after-loop**,
   -fno-rerun-cse-after-loop (default).

The -frerun-cse-after-loop option will re-run common subexpression elimination after loop optimizations has been performed.

Please see *Listing 13* for the C sample code and the compiler generated assembly code with -frerun-cse-after-loop and -fno-rerun-cse-after-loop comparison.

14. **-fcaller-saves**, **-fno-caller-saves** (default).

The -fcaller-saves option enables values to be allocated in registers that will be clobbered by function calls, by emitting extra instructions to save and restore the registers around such calls. Such allocation is done only when it seems the result in better code than would otherwise be produced.

Please see *Listing 14* for the C sample code and the compiler generated assembly code with -fcaller-saves and -fno-caller-saves comparison.

15. **-funroll-loops**, **-fno-unroll-loops** (default).

The -funroll-loops option performs the optimization of loop unrolling. This is only done for loops whose number of iterations can be determined at compile time or run time.

Example:

```
    test()
    {
      int i;
      int j = 10;
      for (i=0;i<5;i++)
      {
        foo();
        j += i;
      }
      return j;
    }
```

The assembly code with -funroll-loops option:

```
_test:
   jsr     @_foo
   jsr     @_foo
   jsr     @_foo
   jsr     @_foo
   jsr     @_foo
   mov.w   #20,r0
   rts
```

The assembly code with -fno-unroll-loops option:

```
_test:
   push    r5
   push    r4
   mov.w   #10,r5
   sub.w   r4,r4
.L5:
   jsr     @_foo
   add.w   r4,r5
   adds    #1,r4
   mov.w   #4,r2
   cmp.w   r2,r4
   ble     .L5
   mov.w   r5,r0
   pop     r4
   pop     r5
   rts
```

Please see *Listing 15* for the C sample code and the compiler generated assembly code with -funroll-loops and -fno-unroll-loops comparison.

16. **-funroll-all-loops**, **-fno-unroll-all-loops** (default).

The -funroll-all-loops option performs the optimization of loop unrolling. This is done for all loops and usually makes programs run more slowly.

Please see *Listing 16* for the C sample code and the compiler generated assembly code with -funroll-all-loops and -fno-unroll-all-loops comparison.

17. **-fshort-enums**, **-fno-short-enums** (default).

The -fshort-enums option allocates to an 'enum' type only as many bytes as it needs for the declared range of possible values. Specifically, the 'enum' type will be equivalent to the smallest integer type which has enough room. The keyword 'enum' is used to declare enumeration types. It provides a means of naming a finite set, and of declaring identifiers as elements of the set.

Example:

```
   enum fruits array[] = {banana,
                 lemon,orange,end};
```

The assembly code with -fshort-enums option:

```
    .byte 3
    .byte 2
    .byte 1
    .byte 0
```

The assembly code with -fno-short-enums option:

```
    .word 3
    .word 2
    .word 1
    .word 0
```

Please see *Listing 17* for the C sample code and the compiler generated assembly code with -fshort-enums and -fno-short-enums comparison.

18. **-fcommon** (default), **-fno-common**.

The -fcommon option allocates uninitialized global variables in the 'common' section. Turning off this option will result in allocating uninitialized global variables in the 'bss' or 'global' section.

Example:

```
    int global_var;
    test()
    {}
```

The assembly code with -fcommon option:

```
    .comm _global_var,2
```

The assembly code with -fno-common option:

```
    .global _global_var,2
```

Please see *Listing 18* for the C sample code and the compiler generated assembly code with -fcommon and -fno-common comparison.

SUMMARY

By default, the compiler will not perform any optimizations. There are some command line switches that the compiler uses to enable optimizations. The switches can be grouped into two groups, which are:

1. **-O**, **-O1**.

Enable these switches will turn on '*-fthread-jumps*','*-fdelayed-branch*', '*-fomit-frame-pointer*', '*-funroll-loops*', and '*-funroll-all-loops*'. We can see the result of optimizations by generating the assembly code. The -S switch can be used to generate the assembly code.

Example:

gcc -S -O Test.c

where

gcc	name of the GNU compiler.
-S	compiler switch to produce assembly code. The generated assembly code will have the extension .s (i.e., Test.s).
-O	compiler switch to perform optimizations.
Test.c	sample file name for input.

2. **-O2**, **-O3**, **-O4**, **-O5**.

Enable these switches will turn on all the -fflag options, except '*-fomit-frame-pointer*', '*-funroll-loops*', and '*-funroll-all-loops*'. We can force the compiler to perform the unperformed -fflag optimizations by enabling the specified -fflag option. We can see the result of optimizations by generating the assembly code. The -S switch can be used to generate the assembly code.

Example:

gcc -S -O5 -fomit-frame-pointer Test.c

where:

-O5	compiler switch to perform optimizations.
-fomit-frame-pointer	compiler switch to perform omit frame pointer optimization.

Listing 1. **Float-store** (begin)

```
/*
 * float-store.c
 * sample code for the following -f options:
 *      -ffloat-store and -fno-float-store
 */

float global_var;

test (char *p)
{
  float k = 0.123456789;

  /* while loop */
  while (*p)
    {
      k++;
    }
  return k;
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and Cygnus
                                                 Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


    .file "float-store.c"                       |  .file "no-float-store.c"
    .section .text                                 .section .text
    .align 2                                       .align 2
    .LC0:                                          .LC0:
    .float 1.23456791043281555176e-1               .float 1.23456791043281555176e-1
    .align 2                                       .align 2
    .LC1:                                          .LC1:
    .float 1.00000000000000000000e0                .float 1.00000000000000000000e0
    .align 2                                       .align 2
    .global _test                                  .global _test
_test:                                          _test:
        mov.w   #6,r3                           <
        sub.w   r3,sp                           <
        push    r5                              <
        push    r4                                 push     r4
        mov.w   r0,@(4,r7)                      |  mov.w    r0,r4
        mov.w   @.LC0,r4                        |  mov.w    @.LC0,r0
        mov.w   @.LC0+2,r5                      |  mov.w    @.LC0+2,r1
        mov.w   r4,@(6,r7)                      |  bra      .L5
        mov.w   r5,@(8,r7)                      <
        mov.b   @r0,r2l                         <
        beq     .L3                             <
.L4:                                            .L4:
        mov.w   @(6,r7),r2                      |  mov.w    @.LC1,r2
        mov.w   @(8,r7),r3                      |  mov.w    @.LC1+2,r3
        push.w      r3                             push.w   r3
        push.w  r2                                 push.w   r2
        mov.w   @.LC1,r0                        <
        mov.w   @.LC1+2,r1                      <
        jsr     @___addsf3                         jsr      @___addsf3
        adds    #2,r7                              adds     #2,r7
        adds    #2,r7                              adds     #2,r7
        mov.w   r0,@(6,r7)                      |.L5:
        mov.w   r1,@(8,r7)                      |  mov.b    @r4,r2l
        mov.w   @(4,r7),r5                      <
        mov.b   @r5,r2l                         <
        bne     .L4                                bne      .L4
.L3:                                            <
        mov.w   @(6,r7),r0                      <
        mov.w   @(8,r7),r1                      <
        jsr     @___fixsfsi                        jsr      @___fixsfsi
        mov.w   r1,r0                              mov.w    r1,r0
        pop     r4                                 pop      r4
        pop     r5                              <
        mov.w   #6,r3                           <
        add.w   r3,sp                           <
        rts                                        rts
        .comm _global_var,4                        .comm _global_var,4
        .end                                       .end
```

Listing 1. **Float-store** (end).

Listing 2. **Defer-pop** (begin).

```
/*
 * defer-pop.c
 * sample code for the following -f options:
 *      -fdefer-pop and -fno-defer-pop
 */

test (char *p)
{

  foo (1, 2, 3, 4, 5);
  foo (1, 2, 3, 4, 5);
  foo (1, 2, 3, 4, 5);
  foo (1, 2, 3, 4, 5);
  foo (1, 2, 3, 4, 5);

}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and Cygnus
                                                 Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5

   .file "defer-pop.c"                          |  .file "no-defer-pop.c"
   .section .text                                  .section .text
   .align 2                                        .align 2
   .global _test                                   .global _test
_test:                                          _test:
   push     r5                                     push     r5
   push     r4                                     push     r4
   mov.w    #5,r5                                  mov.w    #5,r5
   push.w   r5                                     push.w   r5
   mov.w    #4,r4                                  mov.w    #4,r4
   push.w   r4                                     push.w   r4
   mov.w    #3,r2                                  mov.w    #3,r2
   mov.w    #2,r1                                  mov.w    #2,r1
   mov.w    #1,r0                                  mov.w    #1,r0
   jsr      @_foo                                  jsr      @_foo
                                                 > adds     #2,r7
                                                 > adds     #2,r7
   push.w   r5                                     push.w   r5
   push.w   r4                                     push.w   r4
   mov.w    #3,r2                                  mov.w    #3,r2
   mov.w    #2,r1                                  mov.w    #2,r1
   mov.w    #1,r0                                  mov.w    #1,r0
   jsr      @_foo                                  jsr      @_foo
                                                 > adds     #2,r7
                                                 > adds     #2,r7
   push.w   r5                                     push.w   r5
   push.w   r4                                     push.w   r4
   mov.w    #3,r2                                  mov.w    #3,r2
   mov.w    #2,r1                                  mov.w    #2,r1
   mov.w    #1,r0                                  mov.w    #1,r0
   jsr      @_foo                                  jsr      @_foo
                                                 > adds     #2,r7
                                                 > adds     #2,r7
   push.w   r5                                     push.w   r5
   push.w   r4                                     push.w   r4
   mov.w    #3,r2                                  mov.w    #3,r2
   mov.w    #2,r1                                  mov.w    #2,r1
   mov.w    #1,r0                                  mov.w    #1,r0
   jsr      @_foo                                  jsr      @_foo
                                                 > adds     #2,r7
                                                 > adds     #2,r7
   push.w   r5                                     push.w   r5
   push.w   r4                                     push.w   r4
   mov.w    #3,r2                                  mov.w    #3,r2
   mov.w    #2,r1                                  mov.w    #2,r1
   mov.w    #1,r0                                  mov.w    #1,r0
   jsr      @_foo                                  jsr      @_foo
   mov.w    #20,r3                               | adds     #2,r7
   add.w    r3,r7                                | adds     #2,r7
   pop      r4                                     pop      r4
   pop      r5                                     pop      r5
   rts                                             rts
   .end                                            .end
```

Listing 2. **Defer-pop** (end).

Listing 3. **Force-mem** (begin).

```
/*
 * force-mem.c
 * sample code for the following -f options:
 *      -fforce-mem and -fno-force-mem
 */

float global_var;

int k;
int j;

test (char *p)
{

  k += *p;
  j += *p;

}

Differences in assembly code:

;GCC For the Hitachi H8/300                   ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support    ;By Hitachi America Ltd and Cygnus
                                               Support
;release 2.4                                  ;release 2.4
; -O5                                         ; -O5


     .file "force-mem.c"                     |        .file "no-force-mem.c"
     .section .text                                   .section .text
     .align 2                                         .align 2
     .global _test                                    .global _test
_test:                                        _test:
     mov.b    @r0,r2l                                 mov.b @r0,r2l
     mov.b    #0,r2h                                  mov.b #0,r2h
     mov.w    @_k,r3                                  mov.w @_k,r3
     add.w    r2,r3                          |        add.w r3,r2
     mov.w    r3,r2                          <
     mov.w    r2,@_k                                  mov.w r2,@_k
     mov.b    @r0,r2l                                 mov.b @r0,r2l
     mov.b    #0,r2h                                  mov.b #0,r2h
     mov.w    @_j,r3                                  mov.w @_j,r3
     add.w    r2,r3                          |        add.w r3,r2
     mov.w    r3,r2                          <
     mov.w    r2,@_j                                  mov.w r2,@_j
     rts                                              rts
     .comm _global_var,4                              .comm _global_var,4
     .comm _k,2                                       .comm _k,2
     .comm _j,2                                       .comm _j,2
     .end                                             .end
```

Listing 3. **Force-mem** (end).

Listing 4. **Force-addr** (begin).

```
/*
 * force-addr.c
 * sample code for the following -f options:
 *      -fforce-addr and -fno-force-addr
 */
int j;
int k;
foobar (int *a, int b, int c)
{
  j = 1;
  a[b] = c;
  j++;
}
test ()
{
  j = 100;
  foobar (&k, &j - &k, 12345);
}

Differences in assembly code:

;GCC For the Hitachi H8/300                  ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support   ;By Hitachi America Ltd and
                                              Cygnus Support
;release 2.4                                 ;release 2.4
; -O5                                        ; -O5
        .file   "force-addr.c"                |   .file     "no-force-addr.c"
        .section .text                            .section .text
        .align 2                                  .align 2
        .global _foobar                           .global _foobar
_foobar:                                      _foobar:
        push    r4                            |   mov.w     #1,r3
        mov.w   #_j,r3                        |   mov.w     r3,@_j
        mov.w   #1,r4                         <
        mov.w   r4,@r3                        <
        add.w   r1,r1   ; shal.w                  add.w     r1,r1 ; shal.w
        add.w   r0,r1                             add.w     r0,r1
        mov.w   r2,@r1                            mov.w     r2,@r1
        mov.w   #2,r4                         |   mov.w     #2,r3
        mov.w   r4,@r3                        |   mov.w     r3,@_j
        pop     r4                            <
        rts                                       rts
        .align 2                                  .align 2
        .global _test                             .global _test
_test:                                        _test:
        mov.w   #_j,r1                        <
        mov.w   #100,r3                           mov.w     #100,r3
        mov.w   r3,@r1                        |   mov.w     r3,@_j
        mov.w   #_k,r0                            mov.w     #_k,r0
                                              >   mov.w     #_j,r1
        sub.w   r0,r1                             sub.w     r0,r1
        shar r1h        ; shar.w                  shar r1h         ; shar.w
        rotxr r1l       ; end shar.w              rotxr r1l        ; end shar.w
        mov.w   #12345,r2                         mov.w     #12345,r2
        jsr     @_foobar                          jsr @_foobar
        rts                                       rts
        .comm _j,2                                .comm _j,2
        .comm _k,2                                .comm _k,2
        .end                                      .end
```

Listing 4. **Force-addr** (end).

Listing 5. **Omit-frame-pointer** (begin).

```
/*
 * omit-frame-pointer.c
 * sample code for the following -f options:
 *      -fomit-frame-pointer and -fno-omit-frame-pointer
 */

test (char *p)
{
  int k = 0;

  while (*p)
    {
      k++;
    }
  return k;
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


    .file   "omit-frame-pointer.c"              |   .file "no-omit-frame-pointer.c"
    .section .text                                  .section .text
    .align 2                                        .align 2
    .global _test                                   .global _test
_test:                                          _test:
                                                >   push   r6
                                                >   mov.w  r7,r6
    sub.w   r2,r2                                   sub.w  r2,r2
    mov.b   @r0,r0l                                 mov.b  @r0,r0l
    beq     .L3                                     beq    .L3
.L4:                                           .L4:
    adds    #1,r2                                   adds   #1,r2
    cmp.b   #0,r0l                                  cmp.b  #0,r0l
    bne     .L4                                     bne    .L4
.L3:                                           .L3:
    mov.w   r2,r0                                   mov.w  r2,r0
                                                >   pop    r6
    rts                                             rts
    .end                                            .end
```

Listing 5. **Omit-frame-pointer** (end).

Listing 6. **Inline** (begin).

```
#include <math.h>

/*
 * inline.c
 * sample code for the following -f options:
 *      -finline and -fno-inline
 */

inline
foo (int p)
{
  return p * 5;
}

test (char *p)
{
  int k = 100;

  k = foo (k);

  return k;
}

Differences in assembly code:

;GCC For the Hitachi H8/300                       ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support        ;By Hitachi America Ltd and
                                                   Cygnus Support
;release 2.4                                      ;release 2.4
; -O5                                             ; -O5


    .file   "inline.c"                         |      .file "no-inline.c"
    .section .text                                    .section .text
    .align 2                                          .align 2
    .global _foo                                      .global _foo
_foo:                                             _foo:
    mov.w   r0,r2                                     mov.w r0,r2
    add.w r0,r0    ; shal.w                           add.w r0,r0   ; shal.w
    add.w r0,r0    ; shal.w                           add.w r0,r0   ; shal.w
    add.w   r2,r0                                     add.w r2,r0
    rts                                               rts
    .align 2                                          .align 2
    .global _test                                     .global _test
_test:                                            _test:
    mov.w   #500,r0                               |      mov.w #100,r0
                                                  >      jsr   @_foo
    rts                                               rts
    .end                                              .end
```

Listing 6. **Inline** (end).

Listing 7. **Inline-functions** (begin).

```
/*
 * inline-functions.c
 * sample code for the following -f options:
 *      -finline-functions and -fno-inline-functions
 */

int j;
int k;

foobar (int *a, int b, int c)
{
  j = 1;
  a[b] = c;
  j++;
}

test ()
{
  j = 100;
  foobar (&k, &j - &k, 12345);
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5

     .file  "inline-functions.c"                |     .file "no-inline-functions.c"
     .section .text                                   .section .text
     .align 2                                         .align 2
     .global _foobar                                  .global _foobar
_foobar:                                        _foobar:
     mov.w  #1,r3                                     mov.w  #1,r3
     mov.w  r3,@_j                                    mov.w  r3,@_j
     add.w  r1,r1  ; shal.w                           add.w  r1,r1  ; shal.w
     add.w  r0,r1                                     add.w  r0,r1
     mov.w  r2,@r1                                    mov.w  r2,@r1
     mov.w  #2,r3                                     mov.w  #2,r3
     mov.w  r3,@_j                                    mov.w  r3,@_j
     rts                                              rts
     .align 2                                         .align 2
     .global _test                                    .global _test
_test:                                          _test:
     mov.w  #_j,r2                               |    mov.w  #100,r3
     mov.w  #_k,r3                               <
     sub.w  r3,r2                                <
     mov.w  #1,r3                                <
     mov.w  r3,@_j                               <
     and    #254,r2l                             <
     mov.w  #12345,r3                            <
     mov.w  r3,@(_k,r2)                          <
     mov.w  #2,r3                                <
     mov.w  r3,@_j                                    mov.w  r3,@_j
                                                 >    mov.w  #_k,r0
                                                 >    mov.w  #_j,r1
                                                 >    sub.w  r0,r1
                                                 >    shar   r1h   ; shar.w
                                                 >    rotxr  r1l   ; end shar.w
                                                 >    mov.w  #12345,r2
```

```
                                            >     jsr    @_foobar
rts                                               rts
.comm _j,2                                        .comm _j,2
.comm _k,2                                        .comm _k,2
.end                                              .end
```

Listing 7. **Inline-functions** (end).

Listing 8. **Keep-inline-functions** (begin).

```
/*
 * keep-functions-inline.c
 * sample code for the following -f options:
 *      -fkeep-functions-inline and -fno-keep-functions-inline
 */

static inline
foo ()
{
  return 0;
}

test (char *p)
{
  return foo ();
}

Differences in assembly code:

;GCC For the Hitachi H8/300                   ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support    ;By Hitachi America Ltd and
                                               Cygnus Support
;release 2.4                                  ;release 2.4
; -O5                                         ; -O5


     .file "keep-inline-functions.c"          |  .file "no-keep-inline-functions.c"
     .section .text                              .section .text
     .align 2                                    .align 2
_foo:                                         <
     sub.w   r0,r0                            <
     rts                                      <
     .align 2                                 <
     .global _test                               .global _test
_test:                                        _test:
     sub.w   r0,r0                               sub.w   r0,r0
     rts                                         rts
     .end                                        .end
```

Listing 8. **Keep-inline-functions** (end).

Listing 9. **Strength-reduce** (begin).

```
/*
 * strength-reduce.c
 * sample code for the following -f options:
 *      -fstrength-reduce and -fno-strength-reduce
 */

test (short a, int *p, short *s)
{
  int i;
  for (i = 10; i >= 0; i--)
    {
      a = (short) bar ();
      p[i] = a;
      s[i] = a;
    }
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


     .file "strength-reduce.c"                  |     .file "no-strength-reduce.c"
     .section .text                                   .section .text
     .align 2                                         .align 2
     .global _test                                    .global _test
_test:                                          _test:
     subs   #2,sp                                     subs     #2,sp
     push   r5                                        push     r5
     push   r4                                        push     r4
     mov.w  #10,r3                              |     mov.w    r1,@(4,r7)
     mov.w  r2,r5                                     mov.w    r2,r5
     add.b  #20,r5l                             |     mov.w    #10,r4
     addx   #0,r5h                              <
     mov.w  r1,r4                               <
     add.b  #20,r4l                             <
     addx   #0,r4h                              <
.L5:                                            .L5:
     mov.w  r3,@(4,r7)                          <
     jsr    @_bar                                     jsr      @_bar
     mov.w  r0,@r4                              |     mov.w    r4,r3
     mov.w  r0,@r5                              |     add.w    r3,r3   ; shal.w
     subs   #2,r5                               |     mov.w    @(4,r7),r2
     subs   #2,r4                               |     add.w    r3,r2
     mov.w  @(4,r7),r3                          |     mov.w    r0,@r2
     subs   #1,r3                               |     add.w    r5,r3
     mov.w  r3,r3                               |     mov.w    r0,@r3
                                                >     subs     #1,r4
                                                >     mov.w    r4,r4
     bge    .L5                                       bge      .L5
     pop    r4                                        pop      r4
     pop    r5                                        pop      r5
     adds   #2,sp                                     adds     #2,sp
     rts                                              rts
     .end                                             .end
```

Listing 9. **Strength-reduce** (end).

Listing 10. **Thread-jumps** (begin).

```
/*
 * thread-jumps.c
 * sample code for the following -f options:
 *      -fthread-jumps and -fno-thread-jumps
 */

test (char *p)
{
  int i;
  int k;
  int j = 2;
  k = *p;
  if (k)
    {
      j++;

    }
  if (!k)
    {
      k--;
    }

  return k + j;
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


     .file "thread-jumps.c"                     |     .file "no-thread-jumps.c"
     .section .text                                   .section .text
     .align 2                                         .align 2
     .global _test                                    .global _test
_test:                                          _test:
     mov.w    #2,r2                                   mov.w   #2,r2
     mov.b    @r0,r0l                                 mov.b   @r0,r0l
     mov.b    #0,r0h                                  mov.b   #0,r0h
     mov.w    r0,r0                                   mov.w   r0,r0
     beq      .L4                               |     beq     .L2
     mov.w    #3,r2                                   mov.w   #3,r2
     bra      .L3                               |.L2:
.L4:                                            |     mov.w   r0,r0
                                                >     bne     .L3
     mov.w    #-1,r0                                  mov.w   #-1,r0
.L3:                                            .L3:
     add.w    r2,r0                                   add.w   r2,r0
     rts                                              rts
     .end                                             .end
```

Listing 10. **Thread-jumps** (end).

Listing 11. **Cse-follow-jumps** (begin).

```
/*
 * cse-follow-jumps.c
 * sample code for the following -f options:
 *      -fcse-follow-jumps and -fno-cse-follow-jumps
 */

test (char *p)
{
  int a = 0;
  if (*p == a)
    return 0;
  else
    return 1;
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


    .file "cse-follow-jumps.c"                 |      .file "no-cse-follow-jumps.c"
    .section .text                                    .section .text
    .align 2                                          .align 2
    .global _test                                     .global _test
_test:                                         _test:
    mov.b    @r0,r0l                           |      mov.b    @r0,r2l
    mov.b    #0,r0h                            |      mov.b    #0,r2h
    mov.w    r0,r0                             |      mov.w    r2,r2
    beq      .L2                                      beq      .L2
    mov.w    #1,r0                                    mov.w    #1,r0
                                               >      bra      .L4
.L2:                                           .L2:
                                               >      sub.w    r0,r0
                                               >.L4:
    rts                                               rts
    .end                                              .end
```

Listing 11. **Cse-follow-jumps** (end).

Listing 12. **Cse-skip-blocks** (begin).

```
/*
 * cse-skip-blocks.c
 * sample code for the following -f options:
 *      -fcse-skip-blocks and -fno-cse-skip-blocks
 */

test (a, b)
     int *a, b;
{
  if (*a & 123)
    b = 1;
  if (*a & ~222)
    b = 2;
  if (124 & *a)
    b = 3;
  if (~111 & *a)
    b = 4;

  if (~*a & 23)
    b = 1;
  if (~*a & ~22)
    b = 2;
  if (24 & ~*a)
    b = 3;
  if (~11 & ~*a)
    b = 4;

  if (~*a & b)
    b = 1;
  if (~*a & ~b)
    b = 2;
  if (*a & ~*a)
    b = 3;
  return b;
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


    .file   "cse-skip-blocks.c"                 |     .file "no-cse-skip-blocks.c"
    .section .text                                    .section .text
    .align 2                                          .align 2
    .global _test                                     .global _test
_test:                                          _test:
    push    r6                                  |     mov.w   r0,r3
    push    r4                                  <
    mov.w   r0,r6                               <
    mov.w   r1,r0                                     mov.w   r1,r0
    mov.w   r6,r3                               |     mov.w   r3,r2
    adds    #1,r3                               |     adds    #1,r2
    mov.b   @r3,r2l                             |     mov.b   @r2,r2l
    and     #123,r2l                                  and     #123,r2l
    beq     .L2                                       beq     .L2
    mov.w   #1,r0                                     mov.w   #1,r0
.L2:                                            .L2:
    mov.w   @r6,r4                              |     mov.w   @r3,r2
    mov.w   r4,r2                               <
    and     #33,r2l                                   and     #33,r2l
    mov.w   r2,r2                                     mov.w   r2,r2
    beq     .L3                                       beq     .L3
    mov.w   #2,r0                                     mov.w   #2,r0
.L3:                                            .L3:
    mov.b   @r3,r2l                             |     mov.w   r3,r2
                                                >     adds    #1,r2
                                                >     mov.b   @r2,r2l
    and     #124,r2l                                  and     #124,r2l
    beq     .L4                                       beq     .L4
    mov.w   #3,r0                                     mov.w   #3,r0
.L4:                                            .L4:
    mov.w   r4,r2                               |     mov.w   @r3,r2
    and     #144,r2l                                  and     #144,r2l
    mov.w   r2,r2                                     mov.w   r2,r2
    beq     .L5                                       beq     .L5
    mov.w   #4,r0                                     mov.w   #4,r0
.L5:                                            .L5:
    mov.b   @r3,r1l                             |     mov.b   @(1,r3),r2l
    not     r1l                                 |     not     r2l
    mov.b   r1l,r2l                             <
    and     #23,r2l                                   and     #23,r2l
    beq     .L6                                       beq     .L6
    mov.w   #1,r0                                     mov.w   #1,r0
.L6:                                            .L6:
    mov.w   @r6,r3                              |     mov.w   @r3,r2
    not     r3l                                 |     not     r2l
    not     r3h                                 |     not     r2h
    mov.w   r3,r2                               <
    and     #233,r2l                                  and     #233,r2l
    mov.w   r2,r2                                     mov.w   r2,r2
    beq     .L7                                       beq     .L7
    mov.w   #2,r0                                     mov.w   #2,r0
.L7:                                            .L7:
    mov.b   r1l,r2l                             |     mov.b   @(1,r3),r2l
                                                >     not     r2l
    and     #24,r2l                                   and     #24,r2l
    beq     .L8                                       beq     .L8
    mov.w   #3,r0                                     mov.w   #3,r0
.L8:                                            .L8:
    mov.w   r3,r2                               |     mov.w   @r3,r2
                                                >     not     r2l
                                                >     not     r2h
    and     #244,r2l                                  and     #244,r2l
    mov.w   r2,r2                                     mov.w   r2,r2
    beq     .L9                                       beq     .L9
    mov.w   #4,r0                                     mov.w   #4,r0
.L9:                                            .L9:
    mov.w   r3,r2                               |     mov.w   @r3,r2
                                                >     not     r2l
                                                >     not     r2h
    and     r0l,r2l                                   and     r0l,r2l
    and     r0h,r2h                                   and     r0h,r2h
    mov.w   r2,r2                                     mov.w   r2,r2
    beq     .L10                                      beq     .L10
    mov.w   #1,r0                                     mov.w   #1,r0
.L10:                                           .L10:
    mov.w   r0,r2                               |     mov.w   @r3,r2
                                                >     or      r0l,r2l
                                                >     or      r0h,r2h  ;r0 or r2
    not     r2l                                       not     r2l
    not     r2h                                       not     r2h
    and     r3l,r2l                             <
    and     r3h,r2h;                            <
    mov.w   r2,r2                               <
    beq     .L11                                <
    mov.w   #2,r0                               <
.L11:                                           <
    mov.w   r3,r2                               <
    and     r4l,r2l                             <
    and     r4h,r2h                             <
    mov.w   r2,r2                                     mov.w   r2,r2
    beq     .L12                                      beq     .L12
    mov.w   #3,r0                               |     mov.w   #2,r0
.L12:                                           .L12:
    pop     r4                                  <
    pop     r6                                  <
    rts                                               rts
    .end                                              .end
```

Listing 12. **Cse-skip-blocks** (end).

Listing 13. **Rerun-cse-after-loop** (begin).

```
/*
 * rerun-cse-after-loop.c
 * sample code for the following -f options:
 *      -frerun-cse-after-loop and -fno-rerun-cse-after-loop
 */

test (char *p)
{
  int k = 0;

  while (*p)
    {
      k++;
    }
  return k;
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


    .file "rerun-cse-after-loop.c"              |  .file "no-rerun-cse-after-loop.c"
    .section .text                                 .section .text
    .align 2                                       .align 2
    .global _test                                  .global _test
_test:                                          _test:
    sub.w   r2,r2                               |    mov.w  r0,r3
    mov.b   @r0,r0l                             |    sub.w  r0,r0
                                                >    mov.b  @r3,r2l
    beq     .L3                                      beq    .L3
                                                >    mov.b  @r3,r2l
.L4:                                            .L4:
    adds    #1,r2                               |    adds   #1,r0
    cmp.b   #0,r0l                              |    cmp.b  #0,r2l
    bne     .L4                                      bne    .L4
.L3:                                            .L3:
    mov.w   r2,r0                               <
    rts                                              rts
    .end                                             .end
```

Listing 13. **Rerun-cse-after-loop** (end).

Listing 14. **Caller-saves** (begin).

```
/*
 * caller-saves.c
 * sample code for the following -f options:
 *      -fcaller-saves and -fno-caller-saves
 */

test (short a, int *p, short *s)
{
  int i;
  for (i = 10; i >= 0; i--)
    {
      a = (short) bar ();
      p[i] = a;
      s[i] = a;
    }
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


   .file   "caller-saves.c"                     |    .file  "no-caller-saves.c"
   .section .text                                    .section .text
   .align 2                                          .align 2
   .global _test                                     .global _test
_test:                                          _test:
   subs    #2,sp                                     subs   #2,sp
   push    r5                                        push   r5
   push    r4                                        push   r4
   mov.w   #10,r3                                    mov.w  #10,r3
                                                >    mov.w  r3,@(4,r7)
   mov.w   r2,r5                                     mov.w  r2,r5
   add.b   #20,r5l                                   add.b  #20,r5l
   addx    #0,r5h                                    addx   #0,r5h
   mov.w   r1,r4                                     mov.w  r1,r4
   add.b   #20,r4l                                   add.b  #20,r4l
   addx    #0,r4h                                    addx   #0,r4h
.L5:                                            .L5:
   mov.w   r3,@(4,r7)                           <
   jsr     @_bar                                     jsr    @_bar
   mov.w   r0,@r4                                    mov.w  r0,@r4
   mov.w   r0,@r5                                    mov.w  r0,@r5
   subs    #2,r5                                     subs   #2,r5
   subs    #2,r4                                     subs   #2,r4
   mov.w   @(4,r7),r3                                mov.w  @(4,r7),r3
   subs    #1,r3                                     subs   #1,r3
                                                >    mov.w  r3,@(4,r7)
   mov.w   r3,r3                                     mov.w  r3,r3
   bge     .L5                                       bge    .L5
   pop     r4                                        pop    r4
   pop     r5                                        pop    r5
   adds    #2,sp                                     adds   #2,sp
   rts                                               rts
   .end                                              .end
```

Listing 14. **Caller-saves** (end).

Listing 15. **Unroll-loops** (begin).

```
/*
 * unroll-loops.c
 * sample code for the following -f options:
 *      -funroll-loops and -fno-unroll-loops
 */

test ()
{
  int i;
  int j = 10;
  for (i = 0; i < 5; i++)
    {
      foo ();
      j += i;
    }
  return j;
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


   .file   "unroll-loops.c"                     |     .file "no-unroll-loops.c"
   .section .text                                     .section .text
   .align 2                                           .align 2
   .global _test                                      .global _test
_test:                                          _test:
                                                >     push    r5
                                                >     push    r4
                                                >     mov.w   #10,r5
                                                >     sub.w   r4,r4
                                                >.L5:
   jsr     @_foo                                      jsr     @_foo
   jsr     @_foo                                |     add.w   r4,r5
   jsr     @_foo                                |     adds    #1,r4
   jsr     @_foo                                |     mov.w   #4,r2
   jsr     @_foo                                |     cmp.w   r2,r4
   mov.w   #20,r0                               |     ble     .L5
                                                >     mov.w   r5,r0
                                                >     pop     r4
                                                >     pop     r5
   rts                                                rts
   .end                                               .end
```

Listing 15. **Unroll-loops** (end).

Listing 16. **Unroll-all-loops** (begin).

```
/*
 * unroll-all-loops.c
 * sample code for the following -f options:
 *      -funroll-all-loops and -fno-unroll-all-loops
 */
int j;
test (char *p)
{
  int i;
  for (i = 0; i < 5; i++)
    j += 2;
  for (i = 0; i < 100; i++)
    {
      j++;
      a (i);
      if (b (i))
        c (i);
      else
        d (i);
    }
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


   .file  "unroll-all-loops.c"                  |     .file "no-unroll-all-loops.c"
   .section .text                                     .section .text
   .align 2                                           .align 2
   .global _test                                      .global _test
_test:                                          _test:
   push   r5                                     <
   push   r4                                           push     r4
                                                 >     sub.w    r4,r4
   mov.w  #4,r3                                        mov.w    #4,r3
   mov.w  @_j,r2                                 <
   adds   #2,r2                                  <
   mov.w  r2,@_j                                 <
   mov.w  #1,r5                                  <
.L5:                                            .L5:
   mov.w  @_j,r2                                       mov.w    @_j,r2
   add.b  #8,r2l                                 |     adds     #2,r2
   addx   #0,r2h                                 <
   mov.w  r2,@_j                                       mov.w    r2,@_j
   adds   #2,r5                                  |     adds     #1,r4
   adds   #2,r5                                  |     cmp.w    r3,r4
   cmp.w  r3,r5                                  <
   ble    .L5                                          ble      .L5
   sub.w  r5,r5                                  |     sub.w    r4,r4
.L11:                                           .L11:
   mov.w  @_j,r2                                       mov.w    @_j,r2
   adds   #1,r2                                        adds     #1,r2
   mov.w  r2,@_j                                       mov.w    r2,@_j
   mov.w  r5,r0                                  <
   jsr    @_a                                    <
   mov.w  r5,r0                                  <
   jsr    @_b                                    <
   mov.w   r0,r0                                 <
   beq     .L14                                  <
   mov.w   r5,r0                                 <
   jsr     @_c                                   <
   bra     .L13                                  <
.L14:                                            <
   mov.w   r5,r0                                 <
   jsr     @_d                                   <
.L13:                                            <
   mov.w   r5,r4                                 <
   adds    #1,r4                                 <
   mov.w   #99,r2                                <
   cmp.w   r2,r4                                 <
   ble     tl500                                 <
   jmp     @.L12                                 <
   tl500:                                        <
   mov.w   @_j,r2                                <
   adds    #1,r2                                 <
   mov.w   r2,@_j                                <
   mov.w   r4,r0                                 <
   jsr     @_a                                   <
   mov.w   r4,r0                                 <
   jsr     @_b                                   <
   mov.w   r0,r0                                 <
   beq     .L18                                  <
   mov.w   r4,r0                                 <
   jsr     @_c                                   <
   bra     .L17                                  <
.L18:                                            <
   mov.w   r4,r0                                 <
   jsr     @_d                                   <
.L17:                                            <
   mov.w   r5,r4                                 <
   adds    #2,r4                                 <
   mov.w   #99,r2                                <
   cmp.w   r2,r4                                 <
   bgt     .L12                                  <
   mov.w   @_j,r2                                <
   adds    #1,r2                                 <
   mov.w   r2,@_j                                <
   mov.w   r4,r0                                       mov.w   r4,r0
   jsr     @_a                                         jsr     @_a
   mov.w   r4,r0                                       mov.w   r4,r0
   jsr     @_b                                         jsr     @_b
   mov.w   r0,r0                                       mov.w   r0,r0
   beq     .L22                                  |     beq     .L9
   mov.w   r4,r0                                       mov.w   r4,r0
   jsr     @_c                                         jsr     @_c
   bra     .L21                                  |     bra     .L8
.L22:                                            |.L9:
   mov.w   r4,r0                                       mov.w   r4,r0
   jsr     @_d                                         jsr     @_d
.L21:                                            |.L8:
   mov.w   r5,r4                                 <
   adds    #2,r4                                 <
   adds    #1,r4                                       adds    #1,r4
   mov.w   #99,r2                                      mov.w   #99,r2
   cmp.w   r2,r4                                       cmp.w   r2,r4
   bgt     .L12                                  |     ble     .L11
   mov.w   @_j,r2                                <
   adds    #1,r2                                 <
   mov.w   r2,@_j                                <
   mov.w   r4,r0                                 <
   jsr     @_a                                   <
   mov.w   r4,r0                                 <
   jsr     @_b                                   <
   mov.w   r0,r0                                 <
   beq     .L26                                  <
   mov.w   r4,r0                                 <
   jsr     @_c                                   <
   bra     .L25                                  <
.L26:                                             <
   mov.w   r4,r0                                 <
   jsr     @_d                                   <
.L25:                                             <
   adds    #2,r5                                 <
   adds    #2,r5                                 <
   mov.w   #99,r2                                <
   cmp.w   r2,r5                                 <
   bgt     tl501                                 <
   jmp     @.L11                                 <
   tl501:                                        <
.L12:                                             <
   pop     r4                                           pop     r4
   pop     r5                                    <
   rts                                                  rts
   .comm _j,2                                           .comm _j,2
   .end                                                 .end
```

Listing 16. **Unroll-all-loops** (end).

Listing 17. **Short-enums** (begin).

```
/*
 * short-enums.c
 * sample code for the following -f options:
 *      -fshort-enums and -fno-short-enums
 */

enum fruits
  {
   end, orange, lemon, banana
  };

enum fruits array[] =
{banana, lemon, banana, lemon, banana, lemon, orange, lemon, orange, lemon,
orange, lemon, banana, end};

func ()
{
  int citrus = 0;
  enum fruits *p;

  for (p = array; *p != end ; p++)
    if (*p != banana)
      citrus++;
  return citrus;
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


   .file   "short-enums.c"                       |      .file "no-short-enums.c"
   .global _array                                       .global _array
   .section .data                                       .section .data
                                                 >      .align 2
_array:                                          _array:
   .byte 3                                       |      .word 3
   .byte 2                                       |      .word 2
   .byte 3                                       |      .word 3
   .byte 2                                       |      .word 2
   .byte 3                                       |      .word 3
   .byte 2                                       |      .word 2
   .byte 1                                       |      .word 1
   .byte 2                                       |      .word 2
   .byte 1                                       |      .word 1
   .byte 2                                       |      .word 2
   .byte 1                                       |      .word 1
   .byte 2                                       |      .word 2
   .byte 3                                       |      .word 3
   .byte 0                                       |      .word 0
   .section .text                                       .section .text
   .align 2                                             .align 2
   .global _func                                        .global _func
_func:                                           _func:
                                                  >     push      r4
   sub.w   r0,r0                                        sub.w     r0,r0
   mov.w   #_array,r3                                   mov.w     #_array,r3
   bra     .L7                                    |      mov.w     @r3,r4
                                                  >      beq       .L3
                                                  >      mov.w     #3,r1
.L6:                                             .L6:
   mov.b   @r3,r2l                                |      mov.w     @r3,r2
   cmp.b   #3,r2l                                 |      cmp.w     r1,r2
   beq     .L4                                           beq       .L4
   adds    #1,r0                                         adds      #1,r0
.L4:                                             .L4:
   adds    #1,r3                                  |      adds      #2,r3
.L7:                                               |     mov.w     @r3,r4
   mov.b   @r3,r2l                                <
   bne     .L6                                           bne       .L6
                                                  >.L3:
                                                  >      pop       r4
   rts                                                   rts
   .end                                                  .end
```

Listing 17. **Short-enums** (end).

Listing 18. **Common** (begin).

```
/*
 * common.c
 * sample code for the following -f options:
 *      -fcommon and -fno-common
 */

int global_var;

test (char *p)
{
  int k = 0;

  while (*p)
    {
      k++;
    }
  return k;
}

Differences in assembly code:

;GCC For the Hitachi H8/300                     ;GCC For the Hitachi H8/300
;By Hitachi America Ltd and Cygnus Support      ;By Hitachi America Ltd and
                                                 Cygnus Support
;release 2.4                                    ;release 2.4
; -O5                                           ; -O5


   .file   "common.c"                           |       .file "no-common.c"
   .section .text                                       .section .text
   .align 2                                             .align 2
   .global _test                                        .global _test
_test:                                          _test:
   sub.w   r2,r2                                        sub.w     r2,r2
   mov.b   @r0,r0l                                      mov.b     @r0,r0l
   beq     .L3                                          beq       .L3
.L4:                                            .L4:
   adds    #1,r2                                        adds      #1,r2
   cmp.b   #0,r0l                                       cmp.b     #0,r0l
   bne     .L4                                          bne       .L4
.L3:                                            .L3:
   mov.w   r2,r0                                        mov.w     r2,r0
   rts                                                  rts
   .comm _global_var,2                          |       .global _global_var
                                                >       .section .data
                                                >       .align 2
                                                >_global_var:
                                                >       .space 2
   .end                                                 .end
```

Listing 18. **Common** (end).

The information contained in this document has been carefully checked, however the contents of this document may be changed and modified without notice. Hitachi America, Ltd. shall assume no responsibility for inaccuracies, or any problem involving patent infringement caused when applying the descriptions in this document. This material is protected by copyright laws. Hitachi America, Ltd. reserves all rights.

AE-0049 Page 33

HITACHI

Hitachi America, Ltd. • San Francisco Center • 2000 Sierra Point Parkway • Brisbane, CA 94005-1819 • (415) 589-8300