Ruby trunk - Feature #12225

Remove inline assemblers and always enables USE_MACHINE_REGS

03/28/2016 08:20 AM - naruse (Yui NARUSE)

Status:	Rejected
Priority:	Normal
Assignee:	
Target version:	

Description

Current vm_exec.c stores pc an explicitly declared register to get PC.
Since recent CPUs and compilers are very smart, we expect they optimizes their use of registers.

With following patch the benchmark becomes following:

```
diff --git a/vm_exec.c b/vm_exec.c
index 5e4ff94..6f7c1ad 100644
--- a/vm_exec.c
+++ b/vm_exec.c
@@ -15,23 +15,6 @@
 static void vm_analysis_insn(int insn);
 #endif

-#if VMDEBUG > 0
-#define DECL_SC_REG(type, r, reg) register type reg_##r
-
-#elif defined(__GNUC__) && defined(__x86_64__)
-#define DECL_SC_REG(type, r, reg) register type reg_##r __asm__("r" reg)
-
-#elif defined(__GNUC__) && defined(__i386__)
-#define DECL_SC_REG(type, r, reg) register type reg_##r __asm__("e" reg)
-
-#elif defined(__GNUC__) && defined(__powerpc64__)
-#define DECL_SC_REG(type, r, reg) register type reg_##r __asm__("r" reg)
-
-#else
-#define DECL_SC_REG(type, r, reg) register type reg_##r
-#endif
-/* #define DECL_SC_REG(r, reg) VALUE reg_##r */
-
 #if VM_DEBUG_STACKOVERFLOW
 NORETURN(static void vm_stack_overflow_for_insn(void));
 static void
@@ -49,41 +32,12 @@ vm_exec_core(rb_thread_t *th, VALUE initial)
 {

 #if OPT_STACK_CACHING
-#if 0
-#elif __GNUC__ && __x86_64__ && !defined(__native_client__)
-    DECL_SC_REG(VALUE, a, "12");
-    DECL_SC_REG(VALUE, b, "13");
-#else
     register VALUE reg_a;
     register VALUE reg_b;
 #endif
-#endif

-#if defined(__GNUC__) && defined(__i386__)
-    DECL_SC_REG(const VALUE *, pc, "di");
-    DECL_SC_REG(rb_control_frame_t *, cfp, "si");
-#define USE_MACHINE_REGS 1
-
-#elif defined(__GNUC__) && defined(__x86_64__)
-    DECL_SC_REG(const VALUE *, pc, "14");
```

```
-# if defined(__native_client__)
-    DECL_SC_REG(rb_control_frame_t *, cfp, "13");
-# else
-    DECL_SC_REG(rb_control_frame_t *, cfp, "15");
-# endif
-#define USE_MACHINE_REGS 1
-
-#elif defined(__GNUC__) && defined(__powerpc64__)
-    DECL_SC_REG(const VALUE *, pc, "14");
-    DECL_SC_REG(rb_control_frame_t *, cfp, "15");
-#define USE_MACHINE_REGS 1
-
-#else
     register rb_control_frame_t *reg_cfp;
     const VALUE *reg_pc;
-#endif
-
-#if USE_MACHINE_REGS

 #undef  RESTORE_REGS
 #define RESTORE_REGS() \
@@ -98,7 +52,6 @@ vm_exec_core(rb_thread_t *th, VALUE initial)
 #define GET_PC() (reg_pc)
 #undef  SET_PC
 #define SET_PC(x) (reg_cfp->pc = REG_PC = (x))
-#endif

 #if OPT_TOKEN_THREADED_CODE || OPT_DIRECT_THREADED_CODE
 #include "vmtc.inc"

Speedup ratio: compare with the result of `ruby 2.4.0dev (2016-03-27 trunk 54303) [x86_64-linux]'
(greater is better)
name    built-ruby
loop_whileloop       1.016
vm1_attr_ivar*       0.991
vm1_attr_ivar_set*           0.976
vm1_block*           1.013
vm1_const*           0.924
vm1_ensure*          0.978
vm1_float_simple*            1.006
vm1_gc_short_lived*          1.011
vm1_gc_short_with_complex_long*      1.036
vm1_gc_short_with_long*      1.064
vm1_gc_short_with_symbol*            0.997
vm1_gc_wb_ary*       1.005
vm1_gc_wb_ary_promoted*      1.000
vm1_gc_wb_obj*       0.977
vm1_gc_wb_obj_promoted*      1.029
vm1_ivar*            1.054
vm1_ivar_set*        0.961
vm1_length*          1.019
vm1_lvar_init*       0.962
vm1_lvar_set*        0.991
vm1_neq*             0.976
vm1_not*             0.903
vm1_rescue*          0.983
vm1_simplereturn*            1.005
vm1_swap*            1.000
vm1_yield*           0.979

additional example micro benchmark

BEFORE gcc 4.8:
 Performance counter stats for './miniruby -e@v=42; n=100_000_000;while n>0; x=x|x; x=x|x;n-=1;end
':

       7218.124555 task-clock (msec)         #    0.998 CPUs utilized
```

```
               123 context-switches          #    0.017 K/sec
                 2 cpu-migrations            #    0.000 K/sec
               906 page-faults               #    0.126 K/sec
       21374094581 cycles                    #    2.961 GHz
        4469895839 stalled-cycles-frontend   #   20.91% frontend cycles idle
   <not supported> stalled-cycles-backend
       55226298374 instructions              #    2.58  insns per cycle
                                             #    0.08  stalled cycles per insn
        7805291103 branches                  # 1081.346 M/sec
         200172514 branch-misses             #    2.56% of all branches

       7.230608341 seconds time elapsed

BEFORE gcc version 5.3.0 20151204 (Ubuntu 5.3.0-3ubuntu1~14.04):
 Performance counter stats for './miniruby -e@v=42; n=100_000_000;while n>0; x=x|x; x=x|x;n-=1;end
':

       8054.736236 task-clock (msec)         #    0.998 CPUs utilized
               128 context-switches          #    0.016 K/sec
                 2 cpu-migrations            #    0.000 K/sec
               895 page-faults               #    0.111 K/sec
       23776261112 cycles                    #    2.952 GHz
        7078686240 stalled-cycles-frontend   #   29.77% frontend cycles idle
   <not supported> stalled-cycles-backend
       53126508523 instructions              #    2.23  insns per cycle
                                             #    0.13  stalled cycles per insn
        7505454893 branches                  #  931.806 M/sec
         201181233 branch-misses             #    2.68% of all branches

       8.074872624 seconds time elapsed

AFTER gcc version 4.8.5 (Ubuntu 4.8.5-2ubuntu1~14.04.1):

 Performance counter stats for './miniruby -e@v=42; n=100_000_000;while n>0; x=x|x; x=x|x;n-=1;end
':

       7267.867318 task-clock (msec)         #    0.997 CPUs utilized
               169 context-switches          #    0.023 K/sec
                 1 cpu-migrations            #    0.000 K/sec
               899 page-faults               #    0.124 K/sec
       21563673390 cycles                    #    2.967 GHz
        4952119471 stalled-cycles-frontend   #   22.97% frontend cycles idle
   <not supported> stalled-cycles-backend
       53226715304 instructions              #    2.47  insns per cycle
                                             #    0.09  stalled cycles per insn
        7805365852 branches                  # 1073.955 M/sec
         200218594 branch-misses             #    2.57% of all branches

       7.286793973 seconds time elapsed

AFTER gcc version 5.3.0 20151204 (Ubuntu 5.3.0-3ubuntu1~14.04):

 Performance counter stats for './miniruby -e@v=42; n=100_000_000;while n>0; x=x|x; x=x|x;n-=1;end
':

       7146.899779 task-clock (msec)         #    0.998 CPUs utilized
               166 context-switches          #    0.023 K/sec
                 2 cpu-migrations            #    0.000 K/sec
               899 page-faults               #    0.126 K/sec
       21188099959 cycles                    #    2.965 GHz
        4839187155 stalled-cycles-frontend   #   22.84% frontend cycles idle
   <not supported> stalled-cycles-backend
       52525802838 instructions              #    2.48  insns per cycle
                                             #    0.09  stalled cycles per insn
        7505329721 branches                  # 1050.152 M/sec
         200175714 branch-misses             #    2.67% of all branches
```

```
       7.157645157 seconds time elapsed
```

History

#1 - 04/01/2016 10:04 AM - usa (Usaku NAKAMURA)

mswin64 result:

```
Speedup ratio: compare with the result of `ruby 2.4.0dev (2016-04-01 trunk 54468) [x64-mswin64_100]' (greater
is better)
name    built-ruby

loop_whileloop       0.880
vm1_attr_ivar*       0.995
vm1_attr_ivar_set*           0.991
vm1_block*           0.899
vm1_const*           0.890
vm1_ensure*          0.881
vm1_float_simple*            1.044
vm1_gc_short_lived*          0.950
vm1_gc_short_with_complex_long*      0.981
vm1_gc_short_with_long*      0.972
vm1_gc_short_with_symbol*            0.976
vm1_gc_wb_ary*       1.002
vm1_gc_wb_ary_promoted*      0.984
vm1_gc_wb_obj*       1.031
vm1_gc_wb_obj_promoted*      0.955
vm1_ivar*            0.971
vm1_ivar_set*        1.008
vm1_length*          0.881
vm1_lvar_init*       0.967
vm1_lvar_set*        0.859
vm1_neq*             0.899
vm1_not*             0.901
vm1_rescue*          0.970
vm1_simplereturn*            0.945
vm1_swap*            0.806
vm1_yield*           1.020
```

#2 - 04/01/2016 10:49 AM - usa (Usaku NAKAMURA)

One more result of mswin64 (Visual C++ 2013):

```
Speedup ratio: compare with the result of `ruby 2.4.0dev (2016-04-01 trunk 54468) [x64-mswin64_120]' (greater
is better)
name    built-ruby
loop_whileloop       1.058
vm1_attr_ivar*       1.027
vm1_attr_ivar_set*           1.068
vm1_block*           0.964
vm1_const*           0.976
vm1_ensure*          0.710
vm1_float_simple*            1.011
vm1_gc_short_lived*          0.991
vm1_gc_short_with_complex_long*      0.958
vm1_gc_short_with_long*      0.957
vm1_gc_short_with_symbol*            0.971
vm1_gc_wb_ary*       1.036
vm1_gc_wb_ary_promoted*      0.940
vm1_gc_wb_obj*       0.963
vm1_gc_wb_obj_promoted*      1.101
vm1_ivar*            0.945
vm1_ivar_set*        0.980
vm1_length*          0.921
vm1_lvar_init*       1.025
vm1_lvar_set*        1.154
vm1_neq*             0.907
vm1_not*             0.856
vm1_rescue*          0.689
vm1_simplereturn*            0.986
vm1_swap*            1.081
vm1_yield*           1.002
```

#3 - 04/04/2016 05:21 AM - naruse (Yui NARUSE)

- Status changed from Open to Rejected

Usaku NAKAMURA wrote:

> One more result of mswin64 (Visual C++ 2013):
> ...

Thank you for benchmark.
Hmm Visual C++'s optimization is not well developed at 2013 (and Ruby doesn't support VC2015...