Assignment #3 - Global Register Allocation

R04922067 楊翔雲

壹、 問題描述

在 Link Time 進行暫存器重新分配，減少函數需要保存呼叫函數時需要的 push, pop 指令數。

貳、 測試環境

1. Linux 3.19.0-58-generic #64~14.04.1-Ubuntu SMP x86_64 GNU/Linux
2. gcc version 5.3.0 20151204
3. clang++-3.8
4. llvm-3.8.0

參、 算法設計

1. 在 Module Pass 下能見到還沒有轉換成 Machine Code 的每一個 Function，借此可以在全區變數保留 Call Graph。
2. 到 Machine Function Pass 時，所有 Function 已經對應翻成 Machine Function Code，這時暫存器才被決定好，但是跑 Machine Function Pass 順序無法得知。
3. 若能跑第二次 Machine Function Pass，就能記錄每一個 Function 實際使用的暫存器為何，這時候再進行重新指派讓每一個函數之間盡可能使用不同的暫存器，那就可以減少函數之間呼叫時需要的 save/load。
4. 在 Machine Function Pass 處理時，能根據 Machine Function 抓到對應的 IR Function 之間的關係，但這時無法從 IR Function 抓到對應的 Machine Function 的數值。
5. 若函數呼叫為 `A ~> X`，所有任何函數 `A` 若能透過呼叫跑到 `X`，所有 `A` 使用的暫存器集合為 `Reg(set(A))`，將 `X` 分配 `U - Reg(set(A))`。在 DAG 圖上，根據拓樸排序可以做到有效地分配。
6. 在更簡單的操作，維護一個全局暫存器標記，將還沒有使用過的暫存器，重新對應到函數暫存器內使用。

肆、 效能測試 benchmark

1. 以助教給的範例 Test1 為例，在還沒運行撰寫的 Pass，需要 4700012 個指令數。

```
morris1028@miwa ~/L/h/test1> make run PIN_ROOT=../pin-3.0-76991-gcc-linux/
../pin-3.0-76991-gcc-linux//pin -t ./InstCount.so -- ./test1
res = -553279039
time = 0.014
             procedure       calls   self_insts  total_insts
                 _fini           1           3           3
        __libc_csu_fini          0           0           0
        __libc_csu_init          1          34          40
                   run           1      800012     4700012
                     A      100000     1500000     3900000
                     B      100000     1900000     2400000
                     C      100000      500000      500000
              safe_run           1          16     4700028
                  main           1          27     4700091
                    gt           2          36          36
           frame_dummy           1           4           0
  __do_global_dtors_aux          1           8          16
    register_tm_clones           1          12          12
  deregister_tm_clones           1           8           8
                _start           1          11           0
                  .plt           3          17           0
                 _init           1           6           6
```

2. 經由撰寫全局暫存池的重新分配，總指令數量降至 4100010。

```
morris1028@miwa ~/L/h/test1> make run PIN_ROOT=../pin-3.0-76991-gcc-linux/
../pin-3.0-76991-gcc-linux//pin -t ./InstCount.so -- ./test1
res = -553279039
time = 0.013
             procedure       calls   self_insts  total_insts
                 _fini           1            3            3
        __libc_csu_fini          0            0            0
        __libc_csu_init          1           34           40
                   run           1       800010      4100010
                     A      100000      1300000      3300000
                     B      100000      1500000      2000000
                     C      100000       500000       500000
              safe_run           1           16      4100026
                  main           1           27      4100089
                    gt           2           36           36
           frame_dummy           1            4            0
  __do_global_dtors_aux          1            8           16
    register_tm_clones           1           12           12
  deregister_tm_clones           1            8            8
                _start           1           11            0
                  .plt           3           17            0
                 _init           1            6            6
```

3. 在函數 A, B 的紅色部份便是重新分配的暫存器，並把不需要的 push, pop 移除。而在原本的配置，A 原本使用 R14，B 原本使用 R14, R15，將他們分配到所有函數都沒有使用的暫存器 R8, R9, R10，就能減少 push/pop R14/R15 的使用。

```
 27 B:
 28     .cfi_startproc
 29 # BB#0:
 30     pushq   %rbp
 31 .Ltmp3:
 32     .cfi_def_cfa_offset 16
 33 .Ltmp4:
 34     .cfi_offset %rbp, -16
 35     movq    %rsp, %rbp
 36 .Ltmp5:
 37     .cfi_def_cfa_register
%rbp
 38     pushq   %rbx
 39     pushq   %rax
 40 .Ltmp6:
 41     .cfi_offset %rbx, -40
 42 .Ltmp7:
 43     .cfi_offset %r14, -32
 44 .Ltmp8:
 45     .cfi_offset %r15, -24
 46     movl    %edx, %ebx
 47     movl    %esi, %r8d
 48     movl    %edi, %r9d
 49     callq   C
 50     leal    (%r9,%r8), %ecx
 51     addl    %ebx, %ecx
 52     addl    %ecx, %eax
 53     addq    $8, %rsp
 54     popq    %rbx
 55     popq    %rbp
 56     retq
```

```
 64 A:
 65     .cfi_startproc
 66 # BB#0:
 67     pushq   %rbp
 68 .Ltmp9:
 69     .cfi_def_cfa_offset 16
 70 .Ltmp10:
 71     .cfi_offset %rbp, -16
 72     movq    %rsp, %rbp
 73 .Ltmp11:
 74     .cfi_def_cfa_register
%rbp
 75     pushq   %rbx
 76 .Ltmp12:
 77     .cfi_offset %rbx, -32
 78 .Ltmp13:
 79     .cfi_offset %r14, -24
 80     movl    %esi, %ebx
 81     movl    %edi, %r10d
 82     movl    %edx, %edi
 83     movl    %r10d, %edx
 84     callq   B
 85     imull   %r10d, %ebx
 86     addl    %ebx, %eax
 87     popq    %rbx
 88     popq    %rbp
 89     retq
```

伍、 結語

由於在 Machine Function Pass 上獲得的資訊有限，在 LLVM 架構設計下，導致只能用比較髒的寫法，而為了在 lib/Transforms/TestPass/TestPass.cpp 下獲取 X86 指

令和暫存器的資訊，只能複製一份 X86GenInstrInfo.inc / X86GenRegisterInfo.inc / X86GenSubtargetInfo.inc 到 lib/Target/X86/MCTargetDesc/，雖能成功重新配置暫存器，但從大局看來這明顯地會增加編譯時期的記憶體用量以及繁重的配置時間。