ArchLab Solution

姓名：周泽龙
学号：2016013231
课程：计算机与网络体系结构（1）
日期：2018年12月1日

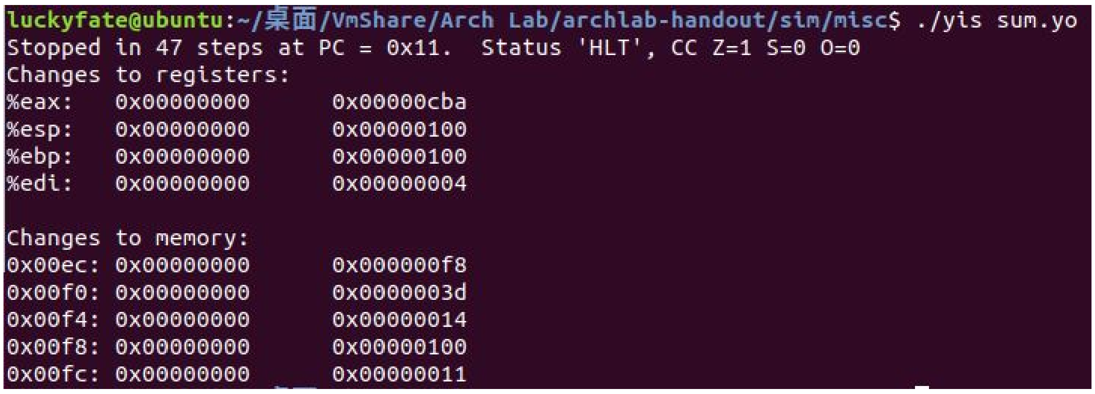
PartA

　　part A要求我们用Y86实现3个C程序，由于汇编课程大作业就做过类似的工作，先用C语言写出了一个音乐播放器，然后在逐语句翻译成汇编代码，因此这部分实验相对来说比较轻松。不同的就是Y86与X86的语法规则略有不同。
　　然后翻译过程中发现实验提供的Y86并不支持寄存器对于立即数的加减，即寄存器无法直接+1或-1，必须通过把立即数1赋给另一个寄存器，然后再对两个寄存器进行操作。这也便是Part B、Part C要求我们实现的功能吧。

sum.ys

```
# 周泽龙，2016013231，软件61
    .pos 0
init:
    irmovl Stack, % esp
    irmovl Stack, % ebp
    call   Main
    halt

# Sample linked list 数据定义
    .align 4
ele1:
    .long 0x00a
    .long ele2
ele2:
    .long 0x0b0
    .long ele3
ele3:
    .long 0xc00
    .long 0

Main:
    pushl   % ebp
    rrmovl  % esp, % ebp
    irmovl  ele1, % edx # edx = ele1
    pushl   % edx
    call    sum_list
    rrmovl  % ebp, % esp
    popl    % ebp
    ret

sum_list:
    pushl   % ebp
    rrmovl  % esp, % ebp
    xorl    % eax, % eax    # 返回值置0  对应C: int val=0
    mrmovl  8(% ebp), % edx  # edx置为表头
    andl    % edx, % edx    # list length == 0 ?
    je  End              # if list length=0, stop.

Loop:
    mrmovl  (% edx), % esi  # esi = ls->val
    addl    % esi, % eax    # 累加 对应C: val += ls->val
    irmovl  $4, % edi
    addl    % edi, % edx
    mrmovl  (% edx), % esi
    rrmovl  % esi, % edx    # edx = ls->next 对应C: ls = ls->next
    andl    % edx, % edx    # ls == 0 ?  对应C: while(ls)
    je  End
    jmp Loop

End:
    rrmovl  % ebp, % esp
    popl    % ebp
    ret


# the stack starts address
    .pos 0x100
Stack:

```

程序运行结果：



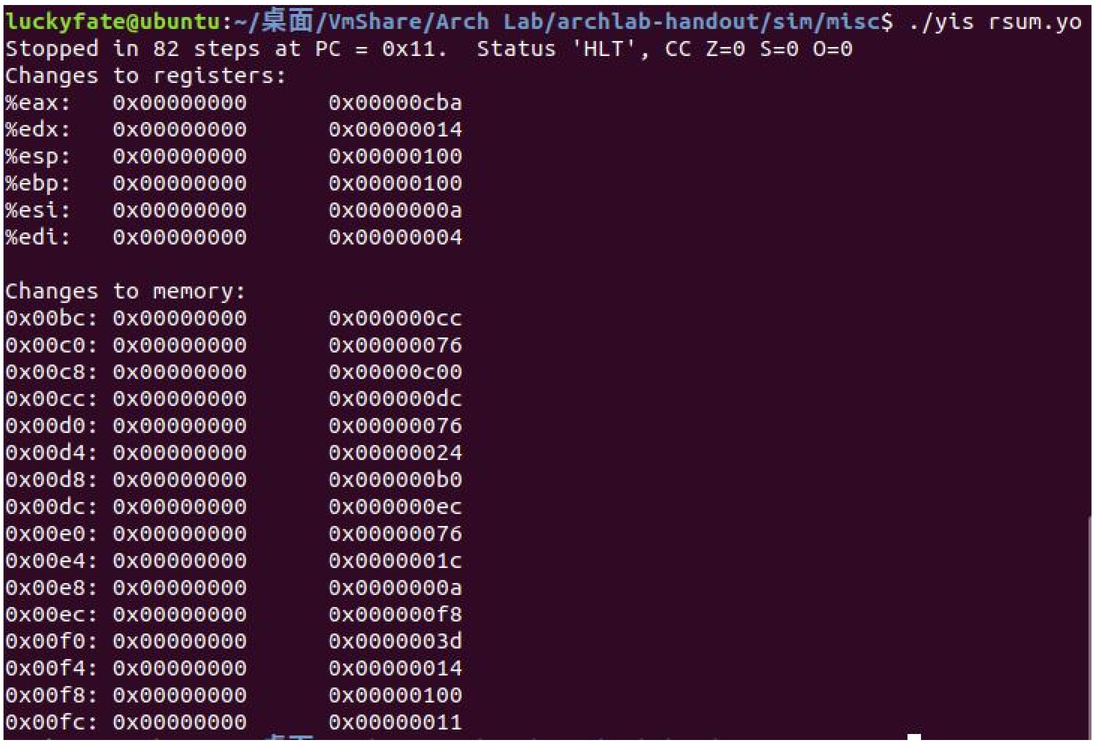
```
luckyfate@ubuntu:~/桌面/VmShare/Arch Lab/archlab-handout/sim/misc$ ./yis sum.yo
Stopped in 47 steps at PC = 0x11.  Status 'HLT', CC Z=1 S=0 O=0
Changes to registers:
%eax:   0x00000000      0x00000cba
%esp:   0x00000000      0x00000100
%ebp:   0x00000000      0x00000100
%edi:   0x00000000      0x00000004

Changes to memory:
0x00ec: 0x00000000      0x000000f8
0x00f0: 0x00000000      0x0000003d
0x00f4: 0x00000000      0x00000014
0x00f8: 0x00000000      0x00000100
0x00fc: 0x00000000      0x00000011
```

rsum.ys

```
# 周泽龙，2016013231，软件61
```

```
    .pos 0
init:
    irmovl Stack, % esp
    irmovl Stack, % ebp
    call   Main
    halt

# Sample linked list 数据定义
    .align 4
ele1:
    .long 0x00a
    .long ele2
ele2:
    .long 0x0b0
    .long ele3
ele3:
    .long 0xc00
    .long 0

Main:
    pushl   % ebp
    rrmovl  % esp, % ebp
    irmovl  ele1, % edx # edx = ele1
    pushl   % edx
    call    rsum_list
    popl    % edx
    rrmovl  % ebp, % esp
    popl    % ebp
    ret

rsum_list:
    pushl   % ebp
    rrmovl  % esp, % ebp
    xorl    % eax, % eax    # 返回值置0 对应C: val=0
    mrmovl  8(% ebp), % edx  # edx置为表头
    andl    % edx, % edx    # list length == 0 ?
    je  End             # if list length=0, stop.

    mrmovl  (% edx), % esi  # esi = ls->val 对应C: ls = ls->next
    pushl   % esi
    irmovl  $4, % edi
    addl    % edi, % edx
    mrmovl  (% edx), % esi
    rrmovl  % esi, % edx    # edx = ls->next 对应C: ls = ls->next
    pushl   % edx
    call    rsum_list  #递归调用 对应C: int rest = rsum_list(ls->next)
    popl    % edx
    popl    % esi  # recover
    addl    % esi, % eax    # 累加 对应C: val += ls->val

End:
    rrmovl  % ebp, % esp
    popl    % ebp
    ret


# the stack starts address
    .pos 0x100
Stack:

```

程序运行结果：



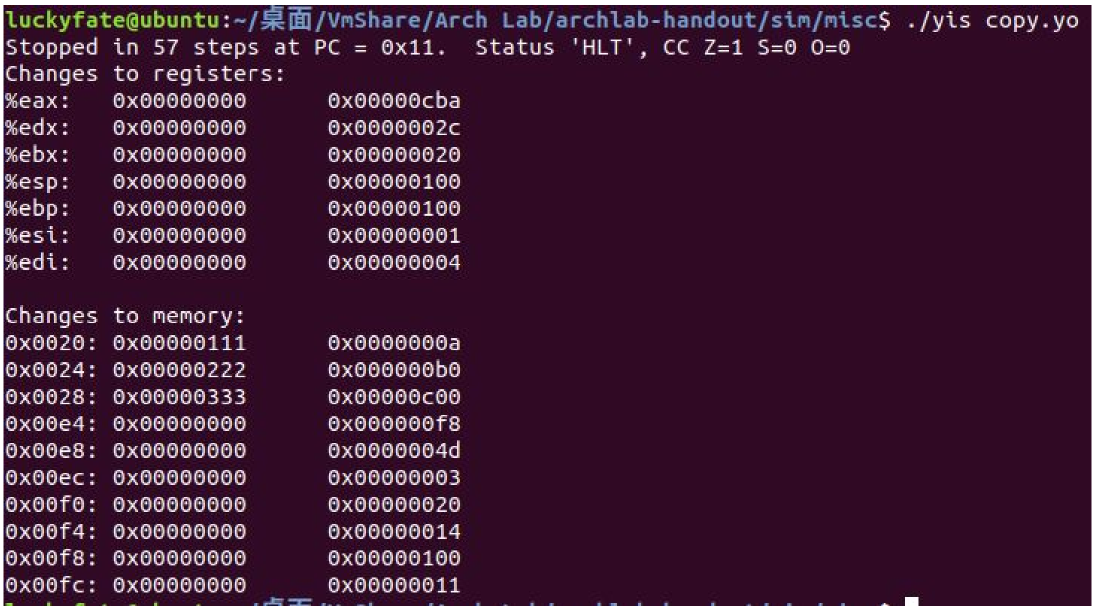
```
luckyfate@ubuntu:~/桌面/VmShare/Arch Lab/archlab-handout/sim/misc$ ./yis rsum.yo
Stopped in 82 steps at PC = 0x11.  Status 'HLT', CC Z=0 S=0 O=0
Changes to registers:
%eax:   0x00000000      0x00000cba
%edx:   0x00000000      0x00000014
%esp:   0x00000000      0x00000100
%ebp:   0x00000000      0x00000100
%esi:   0x00000000      0x0000000a
%edi:   0x00000000      0x00000004

Changes to memory:
0x00bc: 0x00000000      0x000000cc
0x00c0: 0x00000000      0x00000076
0x00c8: 0x00000000      0x00000c00
0x00cc: 0x00000000      0x000000dc
0x00d0: 0x00000000      0x00000076
0x00d4: 0x00000000      0x00000024
0x00d8: 0x00000000      0x000000b0
0x00dc: 0x00000000      0x000000ec
0x00e0: 0x00000000      0x00000076
0x00e4: 0x00000000      0x0000001c
0x00e8: 0x00000000      0x0000000a
0x00ec: 0x00000000      0x000000f8
0x00f0: 0x00000000      0x0000003d
0x00f4: 0x00000000      0x00000014
0x00f8: 0x00000000      0x00000100
0x00fc: 0x00000000      0x00000011
```

copy.ys

```
# 周泽龙，2016013231，软件61
    .pos 0
init:
    irmovl  Stack, % esp
    irmovl  Stack, % ebp
    call    Main
    halt

# Sample linked list 数据定义
    .align 4
src:
    .long 0x00a
    .long 0x0b0
    .long 0xc00
dest:
    .long 0x111
    .long 0x222
    .long 0x333
```

```

Main:
    pushl   % ebp
    rrmovl  % esp, % ebp
    irmovl  src, % edx
    pushl   % edx          # push parameters into stack
    irmovl  dest, % edx
    pushl   % edx          # push parameters into stack
    irmovl  $3, % edx
    pushl   % edx          # push length of list
    call    copy_block
    rrmovl  % ebp, % esp
    popl    % ebp
    ret

copy_block:
    pushl   % ebp
    rrmovl  % esp, % ebp
    xorl    % eax, % eax    # 返回值置0 对应C: val=0
    mrmovl  8(% ebp), % ecx # ecx = length
    mrmovl  12(% ebp), % edx  # edx = dest
    mrmovl  16(% ebp), % ebx  # ebx = src
    andl    % ecx, % ecx    # listlength == 0 ?
    je  End

Loop:
    mrmovl  (% ebx), % esi    # 对应C: int val=src
    rmmovl  % esi, (% edx)  # copy src to dest 对应C: *dest=val
    xorl    % esi, % eax    # 对应C: value ^= val
    irmovl  $4, % edi
    addl    % edi, % ebx    # 对应C: src++
    addl    % edi, % edx    # 对应C: dest++
    irmovl  $1, % esi
    subl    % esi, % ecx    # 对应C: len--
    andl    % ecx, % ecx    # len == 0 ?
    jne Loop

End:
    rrmovl  % ebp, % esp
    popl    % ebp
    ret

# the stack starts address
    .pos 0x100
Stack:

```

程序运行结果：



```
luckyfate@ubuntu:~/桌面/VmShare/Arch Lab/archlab-handout/sim/misc$ ./yis copy.yo
Stopped in 57 steps at PC = 0x11.  Status 'HLT', CC Z=1 S=0 O=0
Changes to registers:
%eax:   0x00000000      0x00000cba
%edx:   0x00000000      0x0000002c
%ebx:   0x00000000      0x00000020
%esp:   0x00000000      0x00000100
%ebp:   0x00000000      0x00000100
%esi:   0x00000000      0x00000001
%edi:   0x00000000      0x00000004

Changes to memory:
0x0020: 0x00000111      0x0000000a
0x0024: 0x00000222      0x000000b0
0x0028: 0x00000333      0x00000c00
0x00e4: 0x00000000      0x000000f8
0x00e8: 0x00000000      0x0000004d
0x00ec: 0x00000000      0x00000003
0x00f0: 0x00000000      0x00000020
0x00f4: 0x00000000      0x00000014
0x00f8: 0x00000000      0x00000100
0x00fc: 0x00000000      0x00000011
```

PartB

在 seq-full.hcl 和 pipe-full.hcl 中添加iaddl和leave指令后，根据实验文档 make VERSION=full ，结果发现在生成的seq-full.c 文件中报错，原因是 I_IADDL 未声明. 需要在../sim/misc/isa.h 头文件中的 itype_t 枚举中添加 I_IADDL 。

iaddl seq

```
iaddl V rB
   fetch:
       icode:ifun <- M1[PC]
       rA:rB <- M1[PC+1]
       valC <- M4[PC+2]
       valP <- PC + 6
   decode:
       valB <- R[rB]
   execute:
       valE <- valB + valC
       set CC
   memory:
   write back:
       R[rB] <- valE
   PC update:
       PC <- valP
```

iaddl pipe

```
iaddl V rB
```

```
    PC selection:
        f_predPC = f_valP
        f_pc = f_predPC
    fetch:
        icode:ifun <- M1[PC]
        rA:rB <- M1[PC+1]
        f_valC <- M4[PC+2]
        f_valP <- PC + 6
    decode:
        d_valB <- R[rb]
    execute:
        e_valE <- valB + valC
    memory:
    write back:
        R[rb] <- W_valE
    PC update:
        PC <- f_valP
```

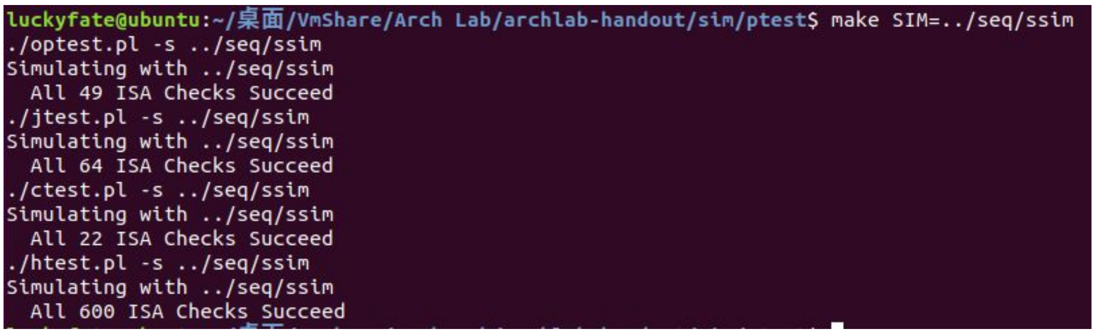
leave seq

```
leave
    fetch:
        icode:ifun <- M1[PC]
        valP <- PC + 1
    decode:
        valA <- R[% esp]
        valB <- R[% ebp]
    execute:
        valE <- valB + 4
    memory:
        valM <- M4[valB]
    write back:
        R[% esp] <- valE
        R[% ebp] <- valM
    PC update:
        PC <- valP
```

leave pipe

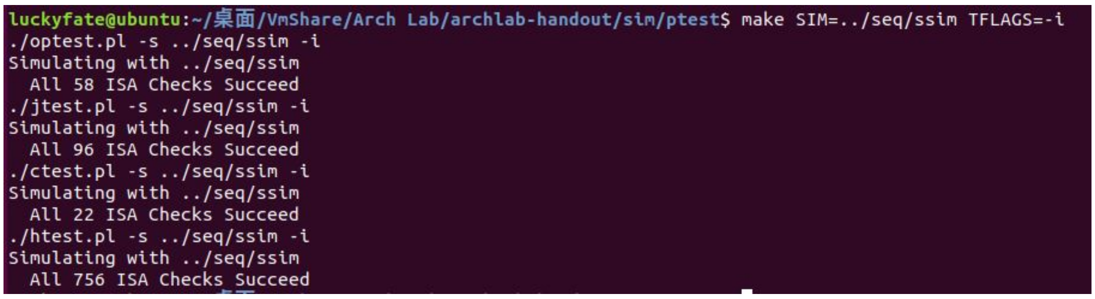
```
leave
    PC selection and fetch:
        icode:ifun <- M1[PC]
        f_valP <- PC + 1
    decode:
        d_valA <- R[% ebp]
        d_valB <- R[% esp]
    execute:
        e_valE <- E_valA + 4
    memory:
        m_valM <- M4[M_valA]
    write back:
        R[% esp] <- W_valE
        R[% ebp] <- W_valM
    PC update:
        PC <- f_valP
```

测试所以指令是否正常（10%）：



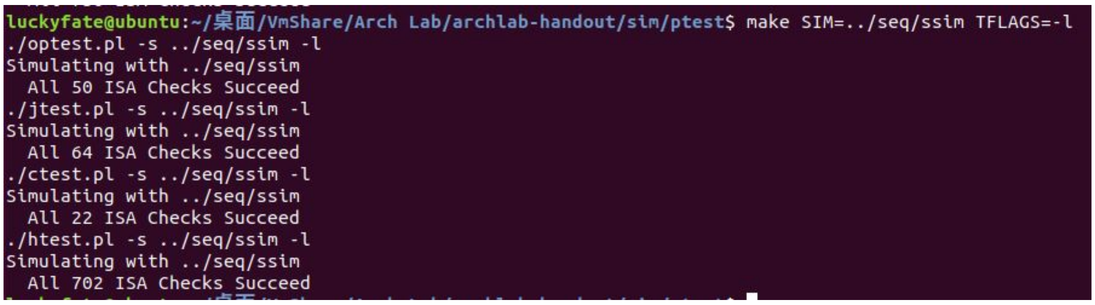
```
luckyfate@ubuntu:~/桌面/VmShare/Arch Lab/archlab-handout/sim/ptest$ make SIM=../seq/ssim
./optest.pl -s ../seq/ssim
Simulating with ../seq/ssim
  All 49 ISA Checks Succeed
./jtest.pl -s ../seq/ssim
Simulating with ../seq/ssim
  All 64 ISA Checks Succeed
./ctest.pl -s ../seq/ssim
Simulating with ../seq/ssim
  All 22 ISA Checks Succeed
./htest.pl -s ../seq/ssim
Simulating with ../seq/ssim
  All 600 ISA Checks Succeed
```

测试iaddl（30%）：



```
luckyfate@ubuntu:~/桌面/VmShare/Arch Lab/archlab-handout/sim/ptest$ make SIM=../seq/ssim TFLAGS=-i
./optest.pl -s ../seq/ssim -i
Simulating with ../seq/ssim
  All 58 ISA Checks Succeed
./jtest.pl -s ../seq/ssim -i
Simulating with ../seq/ssim
  All 96 ISA Checks Succeed
./ctest.pl -s ../seq/ssim -i
Simulating with ../seq/ssim
  All 22 ISA Checks Succeed
./htest.pl -s ../seq/ssim -i
Simulating with ../seq/ssim
  All 756 ISA Checks Succeed
```

测试leave （30%）：



```
luckyfate@ubuntu:~/桌面/VmShare/Arch Lab/archlab-handout/sim/ptest$ make SIM=../seq/ssim TFLAGS=-l
./optest.pl -s ../seq/ssim -l
Simulating with ../seq/ssim
  All 50 ISA Checks Succeed
./jtest.pl -s ../seq/ssim -l
Simulating with ../seq/ssim
  All 64 ISA Checks Succeed
./ctest.pl -s ../seq/ssim -l
Simulating with ../seq/ssim
  All 22 ISA Checks Succeed
./htest.pl -s ../seq/ssim -l
Simulating with ../seq/ssim
  All 702 ISA Checks Succeed
```

Part C

rmxchg

- 作用：交换寄存器和存储器的值
- 用法：rmxchg rA, D(rB)
- 修改过程：
  - 修改 ../misc/yas-grammer.lex，添加 rmxchg 到 Instr 中
  - 修改 ../misc/isa.h，添加 I_RMXCHG 到 itype_t 中
  - 修改 ../misc/isa.c
    - 在 instruction_set[ ] 中添加 {"rmxchg", HPACK(I_RMXCHG, F_NONE), 6, R_ARG, 1, 1, M_ARG, 1, 0}
    - 修改 step_state 函数下的 need_regids 和 need_imm，因为该指令用到了寄存器和立即数

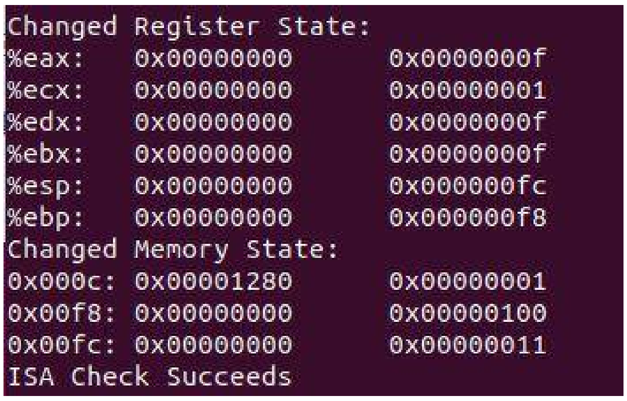
        - 修改step_state 函数下的 switch 语句，加入 case I_RMXCHG
    - 修改 ../pipe/pipe-full.hcl
- 描述

```
rmxchg rA, D(rB)
fetch:
    icode:ifun <- M1[PC]
    rA:rB <- M1[PC + 1]
    valC <- M4[PC + 2]
    valP <- PC + 6
decode:
    valA <- R[rA]
    valB <- R[rB]
 execute:
    valE <- valB + valC
 memory:
    valM <- M4[valE]
    M4[valE] <- valA
 write back:
    R[rA] <- valM
 PC update:
    PC <- valP

```

- 测试代码及解释：
    - 预期结果： eax = 15， ecx = 1

```
# test program for y86 rmxchg
    .pos 0
Init:
    irmovl Stack, % esp
    irmovl Stack, % ebp
    call    Main
    halt

Main:
    pushl % ebp
    rrmovl % esp, % ebp

    irmovl $1, % eax            # eax = 1
    irmovl $15, % ebx           # ebx = 15
    rrmovl % ebx, % edx          # edx = 15
    rmmovl % edx, -3(% ebx)      # save 15(% edx) at address 15-3 = 12
    rmxchg % eax, -3(% ebx)      # exchange the value
    mrmovl -3(% ebx), % ecx      # show the value of address 12
    ret

    .pos 0x100
Stack:

```

    - 测试结果
        - 寄存器 eax 确实与地址12 的存储器交换了值



```
Changed Register State:
%eax:   0x00000000      0x0000000f
%ecx:   0x00000000      0x00000001
%edx:   0x00000000      0x0000000f
%ebx:   0x00000000      0x0000000f
%esp:   0x00000000      0x000000fc
%ebp:   0x00000000      0x000000f8
Changed Memory State:
0x000c: 0x00001280      0x00000001
0x00f8: 0x00000000      0x00000100
0x00fc: 0x00000000      0x00000011
ISA Check Succeeds
```

isubl

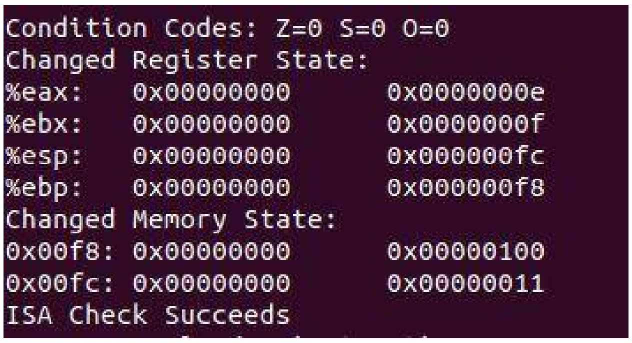
- 作用：寄存器与立即数的减法
- 用法：isubl V rB
- 修改过程：
  - 修改 ../misc/yas-grammer.lex，添加 isubl 到 Instr 中
  - 修改 ../misc/isa.h，添加 I_ISUBL 到 itype_t 中
  - 修改 ../misc/isa.c
    - 在 instruction_set[ ] 中添加 {"isubl", HPACK(I_ISUBL, F_NONE), 6, I_ARG, 2, 4, R_ARG, 1, 0 }
    - 修改 step_state 函数下的 need_regids 和 need_imm，因为该指令用到了寄存器和立即数
    - 修改step_state 函数下的 switch 语句，加入 case I_ISUBL
  - 修改 ../pipe/pipe-full.hcl
  - 修改 ../seq/seq-full.hcl
- 描述

```
isubl V rB
   fetch:
       icode:ifun <- M1[PC]
       rA:rB <- M1[PC+1]
       valC <- M4[PC+2]
       valP <- PC + 6
   decode:
       valB <- R[rB]
   execute:
       valE <- valB - valC
       set CC
   memory:
   write back:
       R[rB] <- valE
   PC update:
       PC <- valP


```

- 测试代码及解释：
  - 预期结果： eax = 14，ebx = 15

```
# test program for y86 isubl
    .pos 0
Init:
    irmovl Stack, % esp
    irmovl Stack, % ebp
    call    Main
    halt

Main:
    pushl % ebp
    rrmovl % esp, % ebp
    irmovl $15, % eax       #eax = 15
    rrmovl % eax, % ebx      #ebx = 15
    isubl $1, % eax         #eax = 14
    ret

    .pos 0x100
Stack:

```

- 测试结果
  - 寄存器 eax 确实减去了1



```
Condition Codes: Z=0 S=0 O=0
Changed Register State:
%eax:   0x00000000      0x0000000e
%ebx:   0x00000000      0x0000000f
%esp:   0x00000000      0x000000fc
%ebp:   0x00000000      0x000000f8
Changed Memory State:
0x00f8: 0x00000000      0x00000100
0x00fc: 0x00000000      0x00000011
ISA Check Succeeds
```