CPU 的应用实践 — 通过递归计算斐波那契数

斐波那契数计算在 32位 Linux 平台下反汇编代码：

```
080483db <fib>:
 80483db:   55                      push   %ebp
 80483dc:   89 e5                   mov    %esp,%ebp
 80483de:   53                      push   %ebx
 80483df:   83 ec 04                sub    $0x4,%esp
 80483e2:   83 7d 08 00             cmpl   $0x0,0x8(%ebp)
 80483e6:   75 07                   jne    80483ef <fib+0x14>
 80483e8:   b8 00 00 00 00          mov    $0x0,%eax
 80483ed:   eb 35                   jmp    8048424 <fib+0x49>
 80483ef:   83 7d 08 01             cmpl   $0x1,0x8(%ebp)
 80483f3:   75 07                   jne    80483fc <fib+0x21>
 80483f5:   b8 01 00 00 00          mov    $0x1,%eax
 80483fa:   eb 28                   jmp    8048424 <fib+0x49>
 80483fc:   8b 45 08                mov    0x8(%ebp),%eax
 80483ff:   83 e8 01                sub    $0x1,%eax
 8048402:   83 ec 0c                sub    $0xc,%esp
 8048405:   50                      push   %eax
 8048406:   e8 d0 ff ff ff          call   80483db <fib>
 804840b:   83 c4 10                add    $0x10,%esp
 804840e:   89 c3                   mov    %eax,%ebx
 8048410:   8b 45 08                mov    0x8(%ebp),%eax
 8048413:   83 e8 02                sub    $0x2,%eax
 8048416:   83 ec 0c                sub    $0xc,%esp
 8048419:   50                      push   %eax
 804841a:   e8 bc ff ff ff          call   80483db <fib>
 804841f:   83 c4 10                add    $0x10,%esp
 8048422:   01 d8                   add    %ebx,%eax
 8048424:   8b 5d fc                mov    -0x4(%ebp),%ebx
 8048427:   c9                      leave
 8048428:   c3                      ret

08048429 <main>:
 8048429:   8d 4c 24 04             lea    0x4(%esp),%ecx
 804842d:   83 e4 f0                and    $0xfffffff0,%esp
 8048430:   ff 71 fc                pushl  -0x4(%ecx)
 8048433:   55                      push   %ebp
 8048434:   89 e5                   mov    %esp,%ebp
 8048436:   51                      push   %ecx
 8048437:   83 ec 14                sub    $0x14,%esp
 804843a:   83 ec 0c                sub    $0xc,%esp
 804843d:   6a 05                   push   $0x5
 804843f:   e8 97 ff ff ff          call   80483db <fib>
 8048444:   83 c4 10                add    $0x10,%esp
 8048447:   89 45 f4                mov    %eax,-0xc(%ebp)
 804844a:   b8 00 00 00 00          mov    $0x0,%eax
 804844f:   8b 4d fc                mov    -0x4(%ebp),%ecx
 8048452:   c9                      leave
```

```
 8048453:    8d 61 fc                lea    -0x4(%ecx),%esp
 8048456:    c3                      ret
```

call 和 ret 指令的实现

以上汇编代码中绝大多数指令在我们实现的 CPU 中只有简化版，因此要用我们的指令集对其进行展开实现。其中重点需要实现的指令有 call 和 ret 。

call 具体功能：

```
decoding.jmp_eip = eip + id_dest->val;
decoding.is_jmp = 1;
rtl_push(decoding.seq_eip);
```

ret 具体功能：

```
rtl_pop(decoding.jmp_eip);
decoding.is_jmp = 1;
```

现在要用我们的指令集实现以上指令重点需要解决将函数调用时的返回地址的压栈出栈问题。

针对这个程序来解决程序调用问题：在 fib 函数 ret 时，有可能跳转到 3 个位置，即 fib 函数里的两处 call 之后和完成斐波那契数计算后返回到 main 函数的 call 之后。

那么就有针对这个程序的实现方法：在三处 call 里替换 push eip 为 push 三个确定的特殊值，在 ret 时根据特殊值来判断跳转到哪个预先设定好的返回地址。

call 的具体实现：

```
#804843f:   e8 97 ff ff ff      call   80483db <fib>
    xor    %temp1,%temp1    01001_100_100_00000
    addi   $7,%temp1         00100_100_00000111
    push   %temp1            10101_100_00000000
    jr     +x                10111_xxx_xxx_xxxxx
```

ret 的具体实现：

```
//#8048428: c3                  ret
    pop    %temp1            10110_100_00000000
    xor    %temp2,%temp2     01001_101_101_00000
    xor    %temp3,%temp3     01001_110_110_00000
    addi   %temp3,$0x7       00100_110_00000111
    cmp    %temp3,%temp1     00101_110_100_00000
    jrz    +x(halt)          11010_000_00000100
    cmp    %temp1,%temp2     00101_100_101_00000
    jrz    +x(up_one)        11010_100_00100010
    jrnz   +x(down_one)      11011_100_00010010
```

其它指令的实现

以上汇编代码用括号的形式隐藏了 load 指令，因此要逐一将他们展开：

```
//#8048430: ff 71 fc            pushl  -0x4(%ecx)
    mov    %temp2,%ecx       01110_101_111_00000
    movil  %temp3,$0x4       01111_110_00000100
    movih  %temp3,$0x0       10000_110_00000000
    sbb    %temp2,%temp3     00001_101_110_00000
    load   %temp1,%temp2     10011_100_101_00000
    push   %temp1             10101_100_00000000
```

注意事项

1. 对我们的 CPU 中的寄存器：r1~r7，分别对应以上汇编代码中的寄存器：eax, ebx, ebp, esp, temp1, temp2, temp3, ecx
2. 以上汇编代码中指令的格式为：op DesR,SourR，我们的 CPU 中的指令格式为 op SR,DR
3. 因为栈是向下增长的，因此初始化时要给 esp 赋一个 RAM 中靠后的位置

```
//初始化：esp 为 512
movih  %esp,$0x02        10000_011_0000_0010
movil  %esp,$0x00        01111_011_0000_0000
```

由以上所有的准备工作得出最终的可以在我们的 CPU 上运行的机器代码（最右侧一列），再存储到 ROM 即可。

```
08048429 <main>:
                                  //初始化：esp 为 512
                                  movih  %esp,$0x02        10000_011_0000_0010
                                  movil  %esp,$0x00        01111_011_0000_0000
 #8048429:  8d 4c 24 04          lea    0x4(%esp),%ecx
                                  mov    %temp1,%esp       01110_100_011_00000
                                  addi   %temp1,$0x4       00100_100_00000100
                                  mov    %ecx,%temp1       01110_111_100_00000
 #804842d:  83 e4 f0             and    $0xfffffff0,%esp
                                  movil  %temp1,$0xf0      01111_100_11110000
                                  movih  %temp1,$0xff      10000_100_11111111
                                  and    %esp,%temp1       00110_011_100_00000
 #8048430:  ff 71 fc             pushl  -0x4(%ecx)
                                  mov    %temp2,%ecx       01110_101_111_00000
                                  movil  %temp3,$0x4       01111_110_00000100

                                  movih  %temp3,$0x0       10000_110_00000000
                                  sbb    %temp2,%temp3     00001_101_110_00000
                                  load   %temp1,%temp2     10011_100_101_00000
                                  push   %temp1             10101_100_00000000
 8048433:   55                   push   %ebp              10101_010_00000000
 #8048434:  89 e5                mov    %esp,%ebp
                                  mov    %ebp,%esp         01110_010_011_00000
```

```
 8048436:   51                    push   %ecx              10101_111_00000000
 #8048437:  83 ec 14              sub    $0x14,%esp
                                    movil  %temp1,$0x14     01111_100_00010100
                                    movih  %temp1,$0x0      10000_100_00000000
                                    sbb    %esp,%temp1      00001_011_100_00000

 #804843a:  83 ec 0c              sub    $0xc,%esp
                                    movil  %temp1,$0xc      01111_100_00001100
                                    movih  %temp1,$0x0      10000_100_00000000
                                    sbb    %esp,%temp1      00001_011_100_00000
 #804843d:  6a 05                 push   $0x2
                                    movil  %temp3,$0x2      01111_110_00000010
                                    movih  %temp3,$0x0      10000_110_00000000
                                    push   %temp3           10101_110_00000000
 #804843f:  e8 97 ff ff ff        call   80483db <fib>
                                    xor    %temp1,%temp1    01001_100_100_00000
                                    addi   $7,%temp1        00100_100_00000111
                                    push   %temp1           10101_100_00000000
                                    jr     +x               10111_xxx_xxx_xxxxx
                                    halt                    11111_000_00000000
                                    see %eax

080483db <fib>:
 80483db:   55                    push   %ebp              10101_010_00000000
 #80483dc:  89 e5                 mov    %esp,%ebp
                                    mov    %ebp,%esp        01110_010_011_00000
 80483de:   53                    push   %ebx              10101_001_00000000
 #80483df:  83 ec 04              sub    $0x4,%esp
                                    movil  %temp1,$0x4      01111_100_00000100
                                    movih  %temp1,$0x0      10000_100_00000000
                                    sbb    %esp,%temp1      00001_011_100_00000
 #80483e2:  83 7d 08 00           cmpl   $0x0,0x8(%ebp)
                                    mov    %temp2,%ebp      01110_101_001_00000
                                    addi   0x8,%temp2       00100_101_00001000
                                    load   %temp1,%temp2    10011_100_101_00000
                                    xor    %temp2,%temp2    01001_101_101_00000

                                    cmp    %temp1,%temp2    00101_100_101_00000
 #80483e6:  75 07                 jne    80483ef <fib+0x14>
                                    jrnz   +x               11011_000_000_00011
 #80483e8:  b8 00 00 00 00        mov    $0x0,%eax
                                    xor    %eax,%eax        01001_000_000_00000
 80483ed:   eb 35                 jmp    8048424 <fib+0x49> 10111_000_00101100
 #80483ef:  83 7d 08 01           cmpl   $0x1,0x8(%ebp)
                                    mov    %temp2,%ebp      01110_101_001_00000
                                    addi   0x8,%temp2       00100_101_00001000
                                    load   %temp1,%temp2    10011_100_101_00000
                                    xor    %temp2,%temp2    01001_101_101_00000
                                    addi   0x1,%temp2       00100_101_00000001
                                    cmp    %temp1,%temp2    00101_100_101_00000

 #80483f3:  75 07                 jne    80483fc <fib+0x21>
                                    jrnz   +x               11011_000_000_00100
 #80483f5:  b8 01 00 00 00        mov    $0x1,%eax
```

```
                                movil  %eax,$0x1         01111_000_00000001
                                movih  %eax,$0x0         10000_000_00000000
80483fa:    eb 28               jmp    8048424<fib+0x49>10111_000_00100010
#80483fc:   8b 45 08            mov    0x8(%ebp),%eax
                                mov    %temp2,%ebp       01110_101_001_00000
                                addi   0x8,%temp2        00100_101_00001000
                                load   %temp1,%temp2     10011_100_101_00000
                                mov    %eax,%temp1       01110_000_100_00000
#80483ff:   83 e8 01            sub    $0x1,%eax
                                movil  %temp1,$0x1       01111_100_00000001
                                movih  %temp1,$0x0       10000_100_00000000

                                sbb    %eax,%temp1       00001_000_100_00000
#8048402:   83 ec 0c            sub    $0xc,%esp
                                movil  %temp1,$0xc       01111_100_00001100
                                movih  %temp1,$0x0       10000_100_00000000
                                sbb    %esp,%temp1       00001_011_100_00000
8048405:    50                  push   %eax              10101_000_00000000
#8048406:   e8 d0 ff ff ff      call   80483db <fib>
                                xor    %temp1,%temp1     01001_100_100_00000
                                push   %temp1             10101_100_00000000
                                jr     +x                 10111_xxx_xxx_xxxxx
#804840b:   83 c4 10            add    $0x10,%esp
                                addi   $0x10,%esp        00100_011_00010000
#804840e:   89 c3               mov    %eax,%ebx
                                mov    %ebx,%eax         01110_001_000_00000

#8048410:   8b 45 08            mov    0x8(%ebp),%eax
                                mov    %temp2,%ebp       01110_101_001_00000
                                addi   0x8,%temp2        00100_101_00001000
                                load   %temp1,%temp2     10011_100_101_00000
                                mov    %eax,%temp1       01110_000_100_00000
#8048413:   83 e8 02            sub    $0x2,%eax
                                movil  %temp1,$0x2       01111_100_00000010
                                movih  %temp1,$0x0       10000_100_00000000
                                sbb    %esp,%temp1       00001_011_100_00000
#8048416:   83 ec 0c            sub    $0xc,%esp
                                movil  %temp1,$0xc       01111_100_00001100
                                movih  %temp1,$0x0       10000_100_00000000
                                sbb    %esp,%temp1       00001_011_100_00000

8048419:    50                  push   %eax              10101_000_00000000
#804841a:   e8 bc ff ff ff      call   80483db <fib>
                                xor    %temp1,%temp1     01001_100_100_00000
                                addi   $2,%temp1         00100_100_00000010
                                push   %temp1             10101_100_00000000
                                jr     +x                 10111_100_00100101
804841f:    83 c4 10            add    $0x10,%esp        00100_011_00010000
#8048422:   01 d8               add    %ebx,%eax
                                add    %eax,%ebx         00000_000_001_00000
#8048424:   8b 5d fc            mov    -0x4(%ebp),%ebx
                                mov    %temp2,%ebp       01110_101_010_00000
                                movil  %temp3,$0x4       01111_110_00000100
                                movih  %temp3,$0x0       10000_110_00000000
```

```
                                sbb    %temp2,%temp3     00001_101_110_00000
                                load   %temp1,%temp2     10011_100_101_00000
                                mov    %ebx,%temp1       01110_001_100_00000
#8048427:  c9                  leave
                                mov    %esp,%ebp         01110_011_010_00000
                                pop    %ebp               10110_010_00000000
#8048428:  c3                  ret
                                pop    %temp1             10110_100_00000000
                                xor    %temp2,%temp2     01001_101_101_00000
                                xor    %temp3,%temp3     01001_110_110_00000
                                addi   %temp3,$0x7       00100_110_00000111
                                cmp    %temp3,%temp1     00101_110_100_00000

                                jrz    +x(halt)          11010_000_00000100
                                cmp    %temp1,%temp2     00101_100_101_00000
                                jrz    +x(up_one)        11010_100_00100010
                                jrnz   +x(down_one)      11011_100_00010010

                                halt                      11111_000_00000000
```