システムプログラムレポート

2024年12月17日
学籍番号：35714121
名前：福富隆大

1. 7-54ページのe7-19.cと7-55ページのe7-20.cをそれぞれgccでコンパイルし，オブジェクトファイルe7-19.oとe7-20.oを作成しなさい

作成したファイル内容、結果について

```
gcc -c e7-1.c
gcc -c e7-3.c
```

上記のコマンドでオブジェクトファイルを作成した。

2. e7-19.oとe7-20.oをそれぞれobjdumpコマンドで逆アセンブルし，gccが生成したアセンブリコードを比較し，その違いを議論しなさい.

実行結果

```
e7-19.o:     ファイル形式 elf64-x86-64


セクション .text の逆アセンブル:

0000000000000000 <sum6>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   c7 05 00 00 00 00 00    movl   $0x0,0x0(%rip)        # 12
<sum6+0x12>
   f:   00 00 00
  12:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 18
<sum6+0x18>
  18:   89 05 00 00 00 00       mov    %eax,0x0(%rip)        # 1e
<sum6+0x1e>
  1e:   eb 23                   jmp    43 <sum6+0x43>
  20:   8b 15 00 00 00 00       mov    0x0(%rip),%edx        # 26
<sum6+0x26>
  26:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 2c
<sum6+0x2c>
  2c:   01 d0                   add    %edx,%eax
  2e:   89 05 00 00 00 00       mov    %eax,0x0(%rip)        # 34
<sum6+0x34>
  34:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 3a
<sum6+0x3a>
```

```
  3a:   83 e8 01                sub    $0x1,%eax
  3d:   89 05 00 00 00 00       mov    %eax,0x0(%rip)        # 43
<sum6+0x43>
  43:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 49
<sum6+0x49>
  49:   85 c0                   test   %eax,%eax
  4b:   7f d3                   jg     20 <sum6+0x20>
  4d:   90                      nop
  4e:   5d                      pop    %rbp
  4f:   c3                      ret
```

```
e7-20.o:     ファイル形式 elf64-x86-64


セクション .text の逆アセンブル:

0000000000000000 <sum7>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   c7 05 00 00 00 00 00    movl   $0x0,0x0(%rip)        # 12
<sum7+0x12>
   f:   00 00 00
  12:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 18
<sum7+0x18>
  18:   89 05 00 00 00 00       mov    %eax,0x0(%rip)        # 1e
<sum7+0x1e>
  1e:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 24
<sum7+0x24>
  24:   85 c0                   test   %eax,%eax
  26:   7e 30                   jle    58 <sum7+0x58>
  28:   90                      nop
  29:   8b 15 00 00 00 00       mov    0x0(%rip),%edx        # 2f
<sum7+0x2f>
  2f:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 35
<sum7+0x35>
  35:   01 d0                   add    %edx,%eax
  37:   89 05 00 00 00 00       mov    %eax,0x0(%rip)        # 3d
<sum7+0x3d>
  3d:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 43
<sum7+0x43>
  43:   83 e8 01                sub    $0x1,%eax
  46:   89 05 00 00 00 00       mov    %eax,0x0(%rip)        # 4c
<sum7+0x4c>
  4c:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 52
<sum7+0x52>
  52:   85 c0                   test   %eax,%eax
  54:   7e 05                   jle    5b <sum7+0x5b>
  56:   eb d1                   jmp    29 <sum7+0x29>
  58:   90                      nop
```

```
    59:   eb 01                   jmp    5c <sum7+0x5c>
    5b:   90                      nop
    5c:   90                      nop
    5d:   5d                      pop    %rbp
    5e:   c3                      ret
```

作成したファイル内容、結果について

e7-19.cでは普通のfor文を使って繰り返しの処理をしており、e7-20.cではifを使って繰り返しの処理をしている。
その結果、e7-19.oではjg命令を使用してループを実装しており、e7-20ではjle命令を使用していた。

アセンブリの命令の数は普通のfor文を使っているe7-19.oの方が少なかった。予想ではe7-20の方がアセンブリに近いので少ない命令になると思っていたので驚いた。 アセンブリをよく見てみるとe7-20の方がnop命令が多く含まれており、コンパイラによる最適化の結果なのだと考えた。

講義に対する感想・質問・意見

while文やfor文をifで書き換える方法はイメージがしやすかったのでとてもすんなり入ってきた。 その際にifの比較の順番を変えることで比較回数を減らすことができる事におどろいた。 if一つをとっても、効率化のための工夫がたくさん行われていることを実感できた。