TÖL309G Tölvutækni og forritun

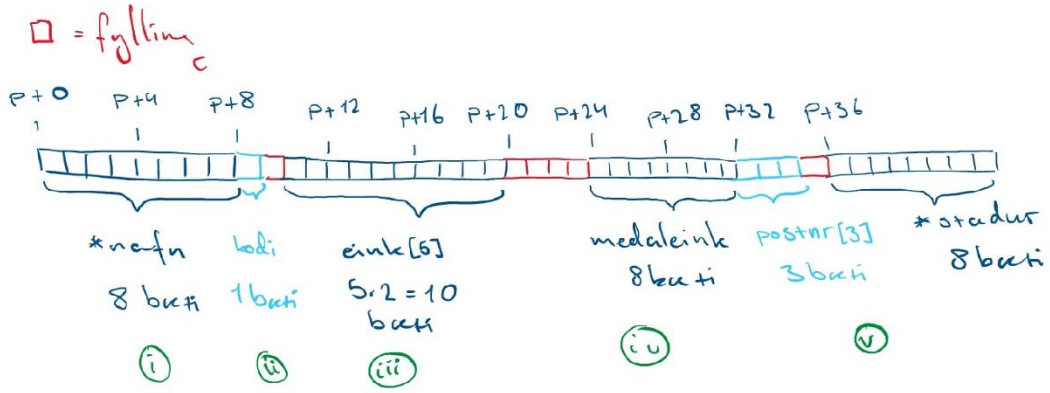
Heimadæmi 8

Hjörvar Sigurðsson

1.

a.

Færslan tekur í heildina 44 bæti.



i. Bendir er 8 bæti.

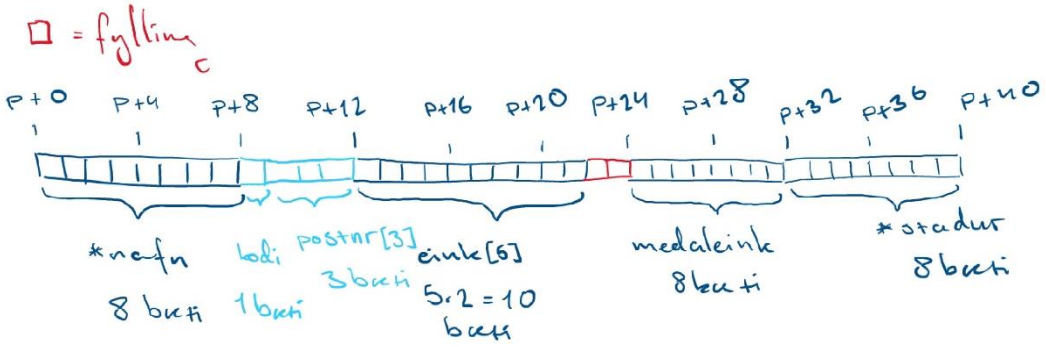
ii. p+8 er margfeldi af stærð char, 1.

iii. Bæta þarf 1-bæta fyllingu aftan á kodi til þess að eink[5] byrji í p+10, sem er margfeldi af stærð short, 2.

iv. Fylling þarf svo medaleink byrji í p+24, sem er margfeldi af stærð double, 8.

v. Fylling þarf svo *stadur byrji í p+36, sem er margfeldi af stærð bendis, 8.

b. Já, það er hægt að umraða sviðum þannig að færslan taki minna minnispláss. Hér tekur hún 40 bæti.



Segjum að i = 2 og j = 4:

i = %rdi = 2

j = %rsi = 4

2.

a. M = 13 og N = 11.

- %rax = (8 * j) = (8 * 4) = 32
- %rax -= %rsi

  %rax = 32 - 4 = 28
- %rax += %rdi

  %rax = 28 + 2 = 30
- %rdx = b + (8 * %rax)

  = b + 240
- %rax = %rdi + 4(%rdi)

  = 2 + 4(2) = 10
- %rax = %rdi + 2(%rax)

  = 2 + 2(10) = 22
- %rsi = %rsi + %rax

  = 4 + 22 = 26
- a + (8(%rsi)) = %rdx

  a + 8(26) = b + 240

  a + 208 = b + 240

208 = i(c)(K) + j(K)

208 = 2(c)(K) + 4K

= 2(C_A)(8) + 4(8)

= 16C_A + 32

=> 176 = 16C_A

C_A = 11

240 = i(c)(K) + j(K)

= 2(c)(K) + 4(K)

= 2(C_B)(8) + 4(8)

= 16C_B + 32

=> 208 = 16C_B

C_B = 13

Eg veit að $Column_A = Row_B$

og $Column_B = Row_A$

Þannig að

a[13][11]

b[11][13].

M = 13 og N = 11

b. Nei, þá hefði ég ekki getað fundið út raðafjölda fylkjanna, eða M. Ég get aðeins fundið út dálkafjölda fylkjanna, eða N, út frá smalamálskóðanum.

3. Á dæmablaði stendur að við sjáum gildið á kanarífuglinum þar sem það er flutt yfir í gistið %rax á ákveðnum tímapunkti. Ég er búinn að rekja mig aftur og aftur í gegnum þetta forrit í gdb í leit að að þessum blessaða fugli, en því miður (að ég held) án árangurs. Ég læt fylgja þær breytingar sem urðu á %eax í gegnum forritið, en %rax var alltaf ‚invalid register' eða ‚void' eftir því hvort ég notaði ‚i r $rax' skipunina eða ‚print $rax'.

   Enn fremur stendur í glærum (og á öllum umræðum um kanarífuglinn sem ég fann á netinu) að samanburðurinn við kanarífuglinn sé á eftirfarandi formi:

   `%fs:40, %rax`

   En ég finn ekkert í þessu forriti, hvorki í gdb leiðangrinum, né í objdump, sem er á þessu formi, þ.e. með ‚:'-færslu.

   <u>Breytingar á %eax:</u>

   Við upphaf forrits er innihald %eax gista 1.

```
(gdb) i r
eax            0x1      1
```

   Í línu 30 í c forritinu, áður en að kallað hefur verið á echo() fallið, þá er innihald %eax gista 14.

```
Type a string:30            echo();
(gdb) i r
eax            0xe      14
```

   Í línu 22 í c forritinu, er kallað á echo() fallið. Í echo fallinu, áður en kallað er á gets() fallið, verður innihald %eax gista 6422268.

```
0x004014bf      22          gets(buf);
(gdb) i r
eax            0x61fefc 6422268
```

   Í línu 10 í c forritinu, í gets() fallinu, er innihald %eax gista 105.

```
i r
0x00401481      10          while ((c = getchar()) != '\n' && c != EOF)
(gdb) i r
eax            0x69     105
```

   Í línu 11 í c forritinu, í gets() fallinu, er innihald %eax gista 6422268.

```
0x00401471      11              *dest++ = c;
(gdb) i r
eax            0x61fefc 6422268
```

   Í línu 10 í c forritinu, í gets() fallinu, er innihald %eax gista 32.

```
(gdb) ni
0x00401481      10          while ((c = getchar()) != '\n' && c != EOF)
(gdb) i r
eax            0x20     32
```

   Í línu 11 í c forritinu, í gets() fallinu, er innihald %eax gista 6422269.

```
0x00401471      11              *dest++ = c;
(gdb) i r
eax            0x61fefd 6422269
```

   Í línu 10 í c forritinu, í gets() fallinu, er innihald %eax gista 114.

```
(gdb) i r
eax            0x72     114
```

Í línu 11 í c forritinu, í gets() fallinu, er innihald %eax gista 6422270.

```
0x00401471      11              *dest++ = c;
(gdb) i r
eax            0x61fefe 6422270
```

Í línu 10 í c forritinu, í gets() fallinu, er innihald %eax gista 10.

```
0x00401481      10          while ((c = getchar()) != '\n' && c != EOF)
(gdb) i r
eax            0xa      10
```

Í línu 14 í c forritinu, í gets() fallinu, er innihald %eax gista 6422271.

```
0x004014a8      14          *dest++ = '\0';         /* Terminate string */
(gdb) i r
eax            0x61feff 6422271
```

Í línu 16 í c forritinu, í gets() fallinu, er innihald %eax gista 6422268.

```
(gdb) i r
eax            0x61fefc 6422268
```

Í línu 24 í c forritinu, í lok echo() fallsins, er innihald %eax gista 0.

4.

a. Það eru tveir minnisaðgangar í lykkjuni í kóðanum.

```
sumv:
        mov     QWORD PTR [rdx], 0
        test    esi, esi
        jle     .L1
        mov     rax, rdi
        lea     ecx, [rsi-1]
        lea     rsi, [rdi+8+rcx*8]
.L3:                                    ← Lykkjan byrjar hér.
        mov     rcx, QWORD PTR [rax]    ← Minnisaðgangur í [rax].
        add     QWORD PTR [rdx], rcx    ← Minnisaðgangur í [rdx].
        add     rax, 8
        cmp     rax, rsi
        jne     .L3
.L1:
        ret
```

b. Í eftirfarandi kóða er aðeins einn minnisaðgangur í lykkjunni. Í stað þess að sækja *s í hverri ítrun lykkjunar, þá er a[0] + a[1] + ... + a[len – 1] lagt saman í lykkjunni, og síðan lagt saman við *s eftir lykkjuna.

<u>C – kóði:</u>

```
void sumv(long *a, int len, long *s) {
    int i;
    long loopSum = 0;
```

```
    *s = 0;
    for (i=0; i<len; i++) {
        loopSum += a[i];
    }

    *s += loopSum;

}
```

Assembly – kóði (þýddur með -O1)

```
sumv:
        mov     QWORD PTR [rdx], 0
        test    esi, esi
        jle     .L4
        mov     rax, rdi
        lea     ecx, [rsi-1]
        lea     rsi, [rdi+8+rcx*8]
        mov     ecx, 0
.L3:                                    ← Lykkjan byrjar hér.
        add     rcx, QWORD PTR [rax]    ← Minnisaðgangur í [rax] hér.
        add     rax, 8
        cmp     rax, rsi
        jne     .L3
.L2:
        mov     QWORD PTR [rdx], rcx
        ret
.L4:
        mov     ecx, 0
        jmp     .L2
```

5.

a. Þetta er dæmi um ‚loop-unrolling‘, eða það að taka aðgerð sem átti að framkvæma í lykkju og fletja hana út, eða framkvæma hana í skrefum án þess að ítra í gegnum lykkju.

C forritið tekur bendi á fylki a, leggur saman fyrstu 10 stök fylkisins og skilar útkomunni.

Smalamálskóðinn (þýddur með -O1 og -funroll-loops):

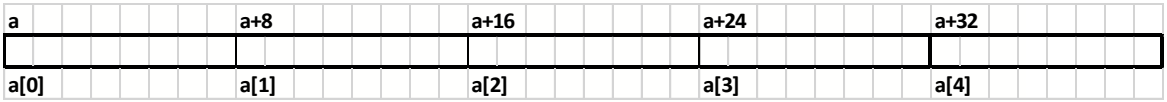
```
sum10:
        mov     rax, QWORD PTR [rdi+8]      i.
        add     rax, QWORD PTR [rdi]        ii.
        add     rax, QWORD PTR [rdi+16]     iii.
        add     rax, QWORD PTR [rdi+24]     iv.
        add     rax, QWORD PTR [rdi+32]     .
        add     rax, QWORD PTR [rdi+40]     .
```

```
        add     rax, QWORD PTR [rdi+48]       .
        add     rax, QWORD PTR [rdi+56]       .
        add     rax, QWORD PTR [rdi+64]       .
        add     rax, QWORD PTR [rdi+72]       v.
        Ret
```

Virkni smalamálskóðans:

Skilagildið, long sum, er geymd í gisti %rax.

i. Fyrst er sum upphafsstillt með a[1].
ii. Næst er framkvæmt samlagninguna sum += a[0].
iii. Næst er framkvæmt samlagninguna sum += a[2].
iv. Næst er framkvæmt samlagninguna sum += a[3].
v. O.s.frv. þar til síðasta samlagningin er sum += a[9], en síðan er skilað sum.



Mynd 5.a. Svona er fylkið a[] í sambandi við smalamálskóðann.

b. Ítrað er í gegnum lykkju .L2 tvisvar sinnum, en í hverri ítrun eru 10 stök fylkisins a.

Smalamálskóðinn (þýddur með -O1 og -funroll-loops):

```
sum10:
        lea     rdx, [rdi+160]                i.
        mov     eax, 0
.L2:
        add     rax, QWORD PTR [rdi]          ii.
        add     rax, QWORD PTR [rdi+8]
        add     rax, QWORD PTR [rdi+16]
        add     rax, QWORD PTR [rdi+24]
        add     rax, QWORD PTR [rdi+32]
        add     rax, QWORD PTR [rdi+40]
        add     rax, QWORD PTR [rdi+48]
        add     rax, QWORD PTR [rdi+56]
        add     rax, QWORD PTR [rdi+64]
        add     rax, QWORD PTR [rdi+72]       iii.
        add     rdi, 80                       iv.
        cmp     rdi, rdx                      v.
        jne     .L2
        ret
```

Virkni smalamálskóðans:

i. Fylkið er 160 / 8 = 20 stök. Síðasta stakið í fylkinu er geymt í gisti %rdx til þess að geta nýst í samanburð seinna.
ii. Framkvæmt er samlagninguna sum = sum (sem er 0 á þessum tímapunkti) + a[0].

iii. Samlagningin sum += a[i] er svo framkvæmd áfram þar til i = 9.

iv. Lagt er 80 við gisti %rdi. Hugsa má um %rdi sem bendi á byrjunarstað í fylkinu fyrir tilkomandi útreikninga. Þetta veldur því að þegar farið er aftur í gegnum .L2, þá byrjar samlagningin (í skrefi ii) í a[10], þar sem 80 / 8 = 10.

v. Samanburður er framkvæmdur á gisti %rdi og %rdx (sjá skref i). Ef að aðeins er búið að leggja saman a[0] + a[1] + ... + a[9], þá er gildið í %rdi 80, og því minna en gisti %rdx, en gisti %rdx = 160, og því aftur farið í gegnum lykkjuna.
Ef búið er að leggja saman a[0] + a[1] + ... + a[19], þá er gildið í %rdi 160, og þá er samlagningu lokið og skilagildinu, sum, er skilað.

c. Fyrst er lagt saman fyrstu þrjú stökin, en næstu 32 eru afgreidd í 4 ítrunum í gegnum lykkju .L2.

Smalamálskóðinn (þýddur með -O1 og -funroll-loops):

```
sum10:
        lea     rcx, [rdi+280]              i.
        mov     rax, QWORD PTR [rdi+8]      ii.
        add     rax, QWORD PTR [rdi]
        add     rax, QWORD PTR [rdi+16]
        lea     rdx, [rdi+24]               iii.
.L2:                                        iv.
        add     rax, QWORD PTR [rdx]
        add     rax, QWORD PTR [rdx+8]
        add     rax, QWORD PTR [rdx+16]
        add     rax, QWORD PTR [rdx+24]
        add     rax, QWORD PTR [rdx+32]
        add     rax, QWORD PTR [rdx+40]
        add     rax, QWORD PTR [rdx+48]
        add     rax, QWORD PTR [rdx+56]
        add     rdx, 64                     v.
        cmp     rdx, rcx                    vi.
        jne     .L2
        ret                                 vii.
```

Virkni smalamálskóðans:

i. Fyrst er geymt vistfang síðasta staks fylkis a í gisti %rcx, svo það nýtist í samanburð seinna.

ii. Skilagildið, sum, er upphafsstillt með a[1], en svo er framkvæmt samlagningarnar sum += a[0] og sum += a[2]. Þá á eftir að bæta stökum a[3] til a[34] við sum.

iii. Hugsa má um %rdx sem bendi á byrjunarstað í fylkinu fyrir tilkomandi útreikninga. Hann er stilltur hér sem a[3].

iv. Í lykkjunni eru framkvæmdar 8x samlagningar, eða sum += a[3] til sum += a[10].
v. Undir lok lykkjunnar er svo bætt 64 við „bendilinn“ okkar, %rdx; það jafngildir því að stilla hann sem a[3+(64/8)] = a[3+8] = a[11].
vi. Borið er saman %rcx (sjá skref i.) og „bendirinn“ okkar, %rdx. Sé aðeins búið að leggja saman fyrstu 13 stök fylkisins a, þá er %rdx = 88, en það er minna en %rcx, sem er 280, og því lykkjan endurtekin.
vii. Þegar farið hefur verið í gegnum lykkjuna fjórum sinnum, þá verður %rdx = 280 þegar að samanburðinum er komið, og skilagildinu, sum, því skilað.