

Tölvutækni og forritun Lokapróf

brj46

Nóvember 2024



Hvað þarf ég að kunna fyrir prófið?

- ☐ Bitavinnsla með heiltölur (signed og unsigned)
- ☐ Einfaldar fleytitölur
- ☐ Skrifa C kóða út frá smalamálskóða (reikniaðgerðir, styriskipanir, föll, bestun,...)
- ☐ Minnisyfirlæði (buffer overflow)
- ☐ Skyndiminni (Skipulag og notkun)
- ☐ Frabrigði, ferlar (fork, wait)
- ☐ Skipulag á sýndarminni
- ☐ Minnisúthlutun

Hlutir til að setja á minnið

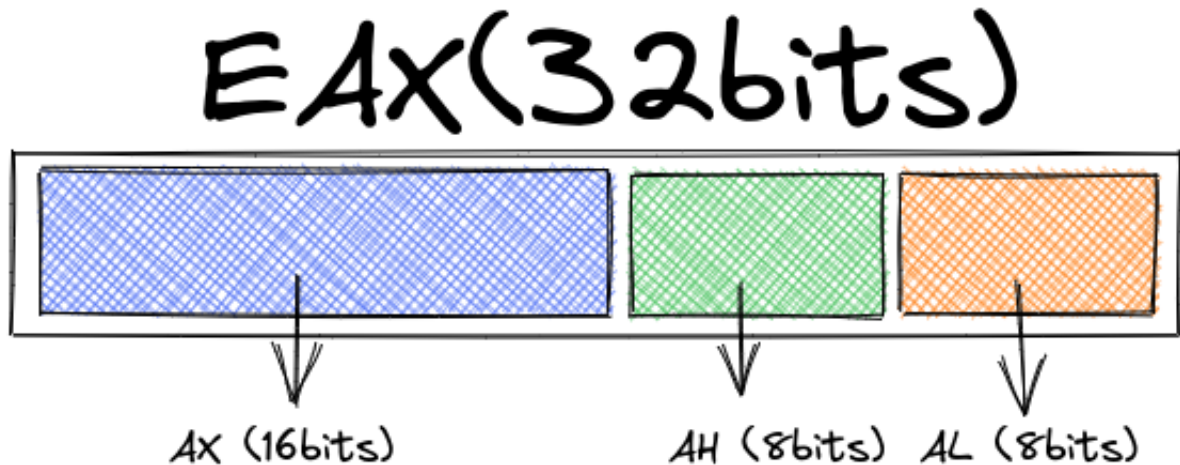
- **mov:** mov (move) flytur gildi á milli minnisstaða, t.d. **movl %eax, %ebx** flytur gildið í **%eax** yfir í **%ebx**
- **lea:** lea (load effective address) er notað til að reikna út minnisstaða og setja hana í register, t.d. **leaq (%rbx, %rcx, 4), %rax** reiknar út **rbx + (rcx * 4)** og setur í **%rax**
- **q:** q stendur fyrir quadword sem er 64 bita gildi
- **l:** l stendur fyrir long sem er 32 bita gildi
- **w:** w stendur fyrir word sem er 16 bita gildi
- **b:** b stendur fyrir byte sem er 8 bita gildi
- **%rax:** 64 bita register
- **%eax:** 32 bita register
- **%ax:** 16 bita register
- **%al:** 8 bita register
- **Diane's Silk Dress Costs 89** (rdi, rsi, rdx, rcx, r8, r9)
- Formúla fyrir vistfang:

$$\text{Vistfanga}[i][j] = \text{Upphafsvistfang} + (i \times \text{fjöldi dalka} + j) \times \text{stærð staks}$$

Tekið frá Tuma

assembly

Uppbrot Gistis



algengar skipanir

| skipun | argument | lýsing |
|------------|----------|--|
| mov | x, y | færir úr x yfir í y, sjá conditional move fyrir neðan |
| push | x | ýtir x á hlaða og eftir að hækka %ESP um sizeof(x) bæti og sett þar inn |
| pop | x | skilar síðasta gildi sem var sett á hlaðann inn í x |
| lea | (x), y | lea, betur þekkt sem leaq er notað til að framkvæma reikning (x) og setja útkomu inn í y |
| (x,y) | | skilar útkomu úr reikningi $x + y$ |
| 0,(x,y) | | skilar útkomu úr reikningi $x * y$ |
| (x,y,z) | | skilar útkomu úr reikningi $x + y * z$ |
| 2(x, y, z) | | skilar útkomu úr reikningi $(2 + (x + y * z))$ |
| sar | x, y | hliðrar y um x bita til hægri, basically heiltöludeiling með x |
| sal | x, y | næstum eins og sar nema til vinstri, núna með margföldun með 2^x |
| sub | x,y | dregur y frá x |
| inc/dec | x | hækkar/lækkar gildi x um 1 |

ath. **SHL** og **SAL** gera það sama en **SHR** virkar ekki með signed int eins og **SAR**

Algeng mynstur

| mynstur | skýring |
|-----------------------|--|
| testl %edi, %edi | logical andað edi við edi þannig ef $edi \leq 0$ er hægt að cmove eða jc í samræmi við það |
| cmove \$5, %eax | færðu 5 inn í eax ef z-flaggið er sett sem 1 þ.e. ef edi er tómt |
| leal 0(%rdi, %rdi, 4) | margfaldar %rdi með 5, $(x + 4 * x)$ |

conditional codes

Þessir kóðar fara a endann á cmov skipunum þ.e. cmov- í línu eftir að eitthvað er testað eins og í dæmi

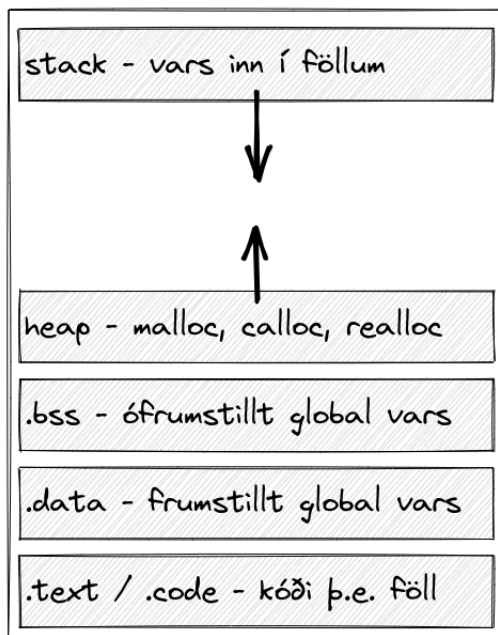
```
testb    $7, %dl
cmov     $1, %rax
```

Þetta er þínu fucked dæmi því e flaggið í cmov stendur fyrir equal nema hvað við erum actually að athuga hvort útkomugildið sé 0, þ.e. að enginn af neðstu 3 bitunum sé 1, þá er gott að muna að e er jafngilt z

Þessi kóði færir 1 inn í %rax ef neðstu þrír bitar %dl eru ekki 111

| cc | condition |
|--------|---------------------------------|
| o | overflow |
| no | no overflow |
| b, nae | below, not above or equal |
| nb, ae | not below, above or equal |
| e, z | equal(zero) |
| ne, nz | not equal, (not zero) |
| na, be | not above, below or equal |
| a, nbe | above, not below or equal |
| s | sign |
| ns | no sign |
| p | parity |
| np | no parity |
| l, nge | less, not greater than or equal |
| nl | not less, greater than or equal |
| ng, le | not greater, less than or equal |
| g, nle | greater, not less than or equal |

minnissvæði



ath. global breytur sem eru skilgreindar sem 0 eða NULL eru líka í .bss

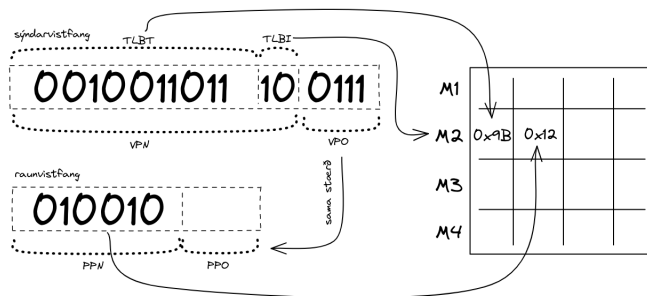
Sýndarminni

- sýndarvístföng: a bitar
- raunvístföng: b bitar
- síðustærð: c bæti
- TLB: d vít, e sæti
- fjöldi mengha: f

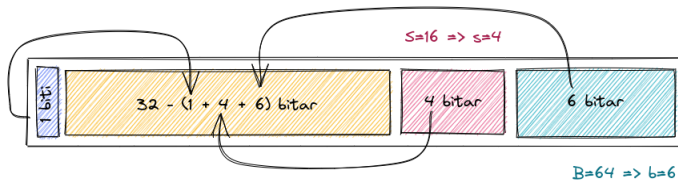
fjöldi mengja er reiknað $\frac{e}{d} = f$

við erum með sýndarvístfang sem er 16 bitar sem skiptast í 4 mengi þá er $\text{VPN}^{\frac{3}{4}} \times 16$ bitar og **VPO** 4 bitar

TBLT og **TLBI** eru skipting á **VPN** og **TBLT** restin raunvístföngin eru jafn löng og **TLBT** og skipt niður í tvo hluta **PPN** og **PPO**, sem er jafn stór og **VPO** (í þessu tilfelli 4 bitar)



annap dæmi, við erum með sýndarminni sem er 4kb að stærð, 4-vítt, E, og með 16 mengi, S, svo út frá þessum tölum finnum við línustærð, B, með reikningnum $\frac{4096}{16 \times 4} = 64$ skiptum þessu nu upp fyrir 32-bitar vístfang:



klukkutífsformúla

$$a + s \times r = m$$

- aðgangstími = a(tif)
- smellahlutfall = s(hlutfall)
- smellarefsing = r (tif)
- meðalaðgangstími = m (tif)

dæmi:

- 97% smellahlutfall, $1 + 0.03 \times 100 = 4$
- 99% smellahlutfall, $1 + 0.01 \times 100 = 2$

Próf 2022 og mínar lausnir við því

1

Í þessu dæmi ætlum við að nota unsigned long breytur til að tákna (allt að) 64 staka mengi (sets). Ef a er unsigned long breyta þá er stak i í menginu a ef biti i ($i = 0, \dots, 63$) er 1, annars er stak i ekki í menginu. Til dæmis væri mengið $\{0, 3\}$, táknað með 64-bita bitastrengnum 00...01001. Athugið að bitarnir eru númeraðir frá hægri til vinstri, svo stak 0 er í menginu, en stak 1 er ekki í menginu, stak 2 er ekki í menginu, o.s.frv.

a.

Skrifið einnar línu fall (þ.e. bara ein **return** skipun) sem skilar sammengi (union) tveggja slíkra mengja. Haus fallsins: **unsigned long sammengi(unsigned long a, unsigned long b)**

Svar:

```
1 unsigned long sammengi(unsigned long a, unsigned long b){
2     return a | b;
3 }
```

b.

Skrifið einnar línu fall sem skilar mengjamun (set difference) mengjanna a og b , þ.e. öll stök sem eru í a , en ekki í b . Haus fallsins: **unsigned long munur(unsigned long a, unsigned long b)**

Svar:

```
1 unsigned long munur(unsigned long a, unsigned long b){
2     return a & ~b;
3 }
```

c.

Athugið að í C er fastinn **1ul** (tölustafurinn 1 og bókstafirnir u og l) 64-bita heiltalan 1 án formerkis. Hvaða mengi táknar segðin ($1ul << i$)?

Svar: Segðin ($1ul << i$) táknar mengið sem inniheldur stakið i og engin önnur stök.

d.

Skrifið einnar línu fall sem skilar því hvort stak i sé í menginu a . Skilagildið á að vera 1 (*satt*) ef i er í a , en 0 (*ósatt*) annars. Þið megið gera ráð fyrir því að gildið á i sé á bilinu 0 til 63. Haus fallsins: **int stakI(unsigned long a, int i)**

Svar:

```
1 int stakI(unsigned long a, int i){
2     return (a >> i) & 1;
3 }
```

2

Við höfum 10-bitu fleytitölur sem fylgja IEEE staðlinum. Við vitum ekki skiptingu þeirra í veldishluta (exp) og brothluta (frac), en það er einn formerkisbiti fremst í þeim.

a.

Hver er lágmarks bitafjöldi í brothluta fleytitölunnar til að hægt sé að tákna töluna $3\frac{9}{16}$ (= 3.5625) nákvæmlega á þessu formi? Rökstyðjið svar ykkar með útreikningi.

Svar: alright við vitum að í IEEE staðlinum erum við með fyrstu töluna sem segir til um + eða mínus næst kemur veldishlutinn og svo brothlutinn.

breytum 3.5625 í binary

$$3.5625 = 11.1001$$

setjum nú töluna í normað form ($1.f \times 2^n$) færum kommuna um einn stað til vinstri 1.11001×2^1 Veldisvísirinn er þá 1 og brothlutinn er 11001

svo Lágmarksbitafjöldi í brothlutanum er því 5 bitar

b.

Hvert er bitagildið fyrir töluna $3\frac{9}{16}$ (= 3.5625) miðað við bitafjöldann fyrir brothlutann sem þið funduð í a-lið (ef þið náðuð ekki að leysa a-liðinn megið þið gefa ykkur raunhæft gildi bitafjöldanum fyrir brothlutann)? Sýnið útreikning á bitagildinu.

Svar:

þar sem við erum að vinna með 10 bitu fleytitölur og við höfum að brothlutinn er 5 bitar þá er veldishlutinn 4 bitar og formerkisbitinn er 1 biti

1. Reiknum bias (skekjustuðulinn)

$$Bias = 2^{4-1} - 1 = 7$$

2. Geymdur veldisvísir er þá

$$GeymtE = E + Bias = 1 + 7 = 8$$

8 í tvíundarkerfi er 1000

3. setjum saman bitana:

- Formeki: 0 (jákvæð tala)
- Veldishlutinn: 1000
- Brothlutinn: 11001

heildarbitaröðin og bitagildið er þá: 0 1000 11001

c.

Segjum að í þessum 10-bita fleytitölum hefur verið ákveðið að nota einn bita fyrir formerki, einn bita fyrir brothluta og restina af bitunum fyrir veldishlutann. Er hægt að tákna staðlaðar tölur á þessu formi, og ef svo er, hver er þá stærsta staðlaða talan sem hægt er að tákna? Rökstyðjið svar ykkar.

Svar: Já við getum táknað staðlaðar tölur á þessu formi

- Formerki: 1 biti
- Veldishlutinn: 8 bitar
- Brothlutinn: 1 biti

Reiknum Bias

$$2^{k-1} - 1 = 2^{8-1} = 2^7 - 1 = 127$$

Hámarks veldistalan er þá $2^{8-1} - 1 = 127$

brothlutinn er 1 og við höfum gefinn 1 bita svo marktalan er 1.1 í binary sem í decimal er 1.5

stærsta staðlaða talan er því

$$1.5 \times 2^{127}$$

d.

Nú hefur verið ákveðið að nota einn bita fyrir formerki, einn bita fyrir veldishluta og restina af bitunum fyrir brothlutann. Er hægt að tákna staðlaðar tölur á þessu formi, og ef svo er, hver er nú stærsta staðlaða talan sem hægt er að tákna? Rökstyðjið svar ykkar.

Svar: Alright reynum þetta þá erum við með

- Formerki: 1 biti
- Veldishlutinn: 1 biti
- Brothlutinn: 8 bitar

reiknum bias

$$2^{k-1} - 1 = 2^{1-1} - 1 = 0$$

Mögulegir veldisvísar eru þá: 0, 1

$$E = 1 - 0 = 1$$

Stærsta talan sem hægt er að tákna er þá:

$$1.1111111_2 \times 2^1 = 1.(2^{1/2^1} + 1/2^2 + 1/2^3 + 1/2^4 + \dots + 2^8) = 1.9960375 \times 2^1$$

Stærsta staðlaða talan:

$$1.9960375 \times 2^1 = 3.9921875$$

3

Hér fyrir neðan er smalamálskóði fallsins `fun`:

```
fun:
    movq    (%rdi), %rax
    jmp     .L2
.L3:
    leaq    (%rax,%rax,2), %rax
    addq    %rax, %rax
    addq    $1, %rsi
.L2:
    cmpq    %rdx, %rsi
    jl      .L3
    ret
```

a.

Hver er fjöldi vistfanga fallsins `fun` og hvert er tag hvers þeirra? Þið eigið að geta séð það út frá notkun gista í kóðanum hér að ofan. Rökstyðjið svarið með vísun í kóðann.

Svar:

Við höfum þrjár inntaksbreytur og eina staðværa breytu

- `%rdi` við sjáum að hér er `movq (rdi), rax` sem bendir til að þetta sé pointer á unsigned long gildi og fyrsta inntaksbreytan
- `%rsi` Notað í samanburði `cmpq rdx, rsi` og síðan uppfærð með `addq $1, rsi` þetta er önnur inntaksbreytan, líklega heiltala.
- `%rdx` Notað í samanburði `cmpq rdx, rsi` þetta er þriðja inntaksbreytan og sennilega líka heiltala
- `%rax` Notað til að geyma gildi sem er lesið úr minni og síðan uppfært í lykkju.

Því erum við með 4 vistföng

1. **Bendir:** unsigned long * (í `%rdi`)
2. **Heiltala s:** unsigned long (í `%rsi`)
3. **Heiltala x:** unsigned long (í `%rdx`)
4. **Staðvær heiltala rax:** unsigned long (í `%rax`)

b.

Skrifið jafngildan C kóða fyrir fallið `fun`. Þið megið velja breytunöfnin, eða nefna þau eftir gistunum.

Svar:

```
1 unsigned long fun(unsigned long *ptr, unsigned long s, unsigned long x){
2     unsigned long rax = *ptr;
3     while(s < x){
4         rax = 6 * rax;
5         s = s + 1;
6     }
7     return rax;
8 }
```

c.

Smalamálskóðinn að ofan er úttak úr þýðandanum **gcc** með bestunarroffann **-Og**. Ef notaður er bestunarroffinn **-O3** (sem er mesta mögulega bestun) fæst kóðinn sem hér er fyrir neðan. Berið hann sama við fyrri kóðann og segið í hvaða tilvikum **O3**-kóðinn gæti verið hraðvirkari.

```
fun:
    movq    (%rdi), %rax
    cmpq    %rdx, %rsi
    jge     .L1
.L3:
    leaq    (%rax,%rax,2), %rax
    addq    %rax, %rax
    addq    $1, %rsi
    cmpq    %rsi, %rdx
    jne     .L3
.L1:
    ret
```

Svar: Byrjum á að skoða munin á milli þessara kóða :

Við sjáum að í seinni kóðanum höfum við strax compare skipun í annari línu sem segir `jge` (jump greater or equal) sem þýðir að ef **rsi** er stærra eða jafnt og **rdx** þá er haldið áfram í **.L1** sem fer beint í return þannig ef **rsi** er minna er **rdx** þá er farið í **.L3**

Næst fáum við `leaq` skipunina sem margfaldar **rax** með 3 og bætir í **rax** síðan er bætt **rax** við **rax** svo í raun er verið að gera $6 \times rax$ síðan er 1 bætt við **rsi** og kannað hvort **rsi** sé orðið jafnt og **rdx** og ef ekki er farið aftur í **.L3** sem myndar lykkju.

Ástæðan af hverju þetta er hraðvirkara er að það í seinna er færri hopp á milli.

4

Hér fyrir neðan er smalamálskóði fyrir endurkvæmt fall með hausinn: **int rec(int n, int m)**:

```
rec:
    cmpl    %esi, %edi    # lína 1
    jge     .L8           # lína 2
    movl    $0, %eax      # lína 3
    ret     # lína 4
.L8:
    subq    $8, %rsp      # lína 5
    addl    %esi, %esi     # lína 6
    subl    $2, %edi       # lína 7
    call    rec           # lína 8
    addl    $1, %eax       # lína 9
    addq    $8, %rsp       # lína 10
    ret     # lína 11
```

a.

Sýnið jafngilt endurkvæmt C fall. Rökstyðjið einstakar skipanir út frá línunum í smalamál-skóðanum.

Svar:

Byrjum á að skoða kóðann:

- lína 1 og 2: ef edi er stærra eða jafnt og esi þá er hoppað í .L8
- 3 og 4: ef edi er minna en esi þá er eax núllstillt og return skipun framkvæmd.
- lína 5: rsp er minnkað um 8
- lína 6: esi er tvöfaldast
- lína 7: edi er minnkað um
- lína 8: kallað er á fallið rec
- lína 9: eax er hækkað um 1
- lína 10: rsp er hækkað um 8
- lína 11: return skipun

alright reynum að átta okkur á breytunum

- edi: við höfum cmp(l) l tekur 32 bita sem þýðir að þetta er líklega heiltala
- esi: við höfum cmp(l) sem þýðir að þetta er líklega heiltala
- eax: við höfum mov(l) skipun sem þýðir að þetta er líklega heiltala
- rsp: við höfum sub(q) q tekur 64 bita sem þýðir að þetta er líklega minnisvísir

```
1  int rec(int n, int m){
2      if(n >= m){
3          m = m + m;
4          n = n - 2;
5          int result = rec(n, m);
6          return result + 1;
7      }
```

```
8         else{
9             return 0;
10        }
11    }
```

b.

Teiknið upp hlaðaramma (stack frame) fallsins **rec** og merkið inn einstök svæði í honum.

Svar:

| Endurkomuvistfang |
|-------------------|
| Ónotað pláss |

5

Þrjár sjálfstæðar spurningar

a.

Gefið er fylkið **short int a[6][10]**. Ef upphafsvistfang þess er 0, hvert er þá vistfang staksins **a[3][4]**? Sýnið útreikning.

Svar:

notum víst þessa formúlu fyrir vistfang

$$\text{Vistfang}a[i][j] = \text{Upphafsvistfang} + (i \times \text{fjöldi dalka} + j) \times \text{stærð staks}$$

- **Upphafsvistfang:** 0
- **fjöldi dalka:** 10
- **stærð staks:** 2 (short int) = 2 bæti

Reiknum:

$$\text{Vistfang}a[3][4] = 0 + (3 \times 10 + 4) \times 2 = 0 + 34 \times 2 = 68$$

Vistfang staksins a[3][4] er því 68

b.

Gefin er eftirfarandi færsla í C forriti:

```
struct abcd {  
    short int a[2];  
    double b;  
    char c;  
    char *d;  
};
```

Rissið upp mynd af henni út frá uppröðunarkröfu (*alignment requirements*) x86-64. Hversu mörg bæti tekur færslan?

Svar:

Skoðum fyrst hvað hvert stak tekur af minni

- Short int a[2]: 2 bæti
- Double b: 8 bæti
- Char c: 1 bæti
- Char *d: 8 bæti

| Offset | Biti | Lýsing |
|--------|------------------|-----------------------------------|
| 0 | a[0] (2bæti) | fyrsta stak í a |
| 2 | a[1] (2bæti) | næsta stak í a |
| 4-7 | Padding (4 bæti) | Fylling til að halda rétttri röð? |
| 8-15 | b (8 bæti) | double b |
| 16 | c (1 bæti) | char c |
| 17-23 | Padding (7 bæti) | Fylling til að halda rétttri röð? |
| 24-31 | d (8 bæti) | char *d |

Færslan tekur því 32 bæti

c.

Skyndiminni er 2-vítt (*2-way set associative*) með heildarstærðina 2048 (=211) bæti og 16 (=24) bæta línur (þ.e. blokkir).

i. Hvað hefur skyndiminnið mörg mengi?

Svar: Til að finna Mengin skulum við nota formúluna

$$\text{Fjöldi Mengi} = \frac{\text{Fjöldi lína}}{\text{Samstillingarstuðull}}$$

Til að finna heildarfjölda lína í skyndiminni er notað formúluna:

$$\text{Fjöldi lína} = \frac{\text{Heildarstærð}}{\text{Línustærð}} = \frac{2048\text{bæti}}{16\text{bæti}} = 128\text{línur}$$

Fjöldi mengja er þá (samstillingarstuðulinn er 2 af því þetta er 2-vítt skyndiminni):

$$\frac{128\text{línur}}{2} = 64\text{mengi}$$

ii. Ef vistföng eru 16 bitar, hversu margir bitar eru þá notaðir fyrir merkið (*tag*)?

Svar:

Til að finna fjölda merkisbita þurfum við að :

1. finna fjölda bita fyrir block Offset
2. finna fjölda bita fyrir set index
3. Draga þessa bita frá heildarfjölda ita í vistfangi til að fá fjölda merkisbita.

Finnur fjölda bita fyrir block Offset:

$$\text{Block offset} = \log_2(\text{Línustærð}) = \log_2(16) = 4\text{bitar}$$

Finnur fjölda bita fyrir set index:

fjöldi mengja er 64 þannig að:

$$\text{Set index} = \log_2(\text{Fjöldi mengja}) = \log_2(64) = 6\text{bitar}$$

Fjöldi merkisbita:

$$\text{Fjöldi merkisbita} = 16 - 4 - 6 = 6\text{bitar}$$

Merkisbitar eru því 6 bitar

6

Hér fyrir neðan er forrit sem býr til ný ferli með fork-fallinu.

```
void fall() {
    if (fork() != 0) {
        fork();
        printf("B\n");
    }
}

int main() {
    printf("A\n");
    fall();
    printf("C\n");
    exit(0);
}
```

a.

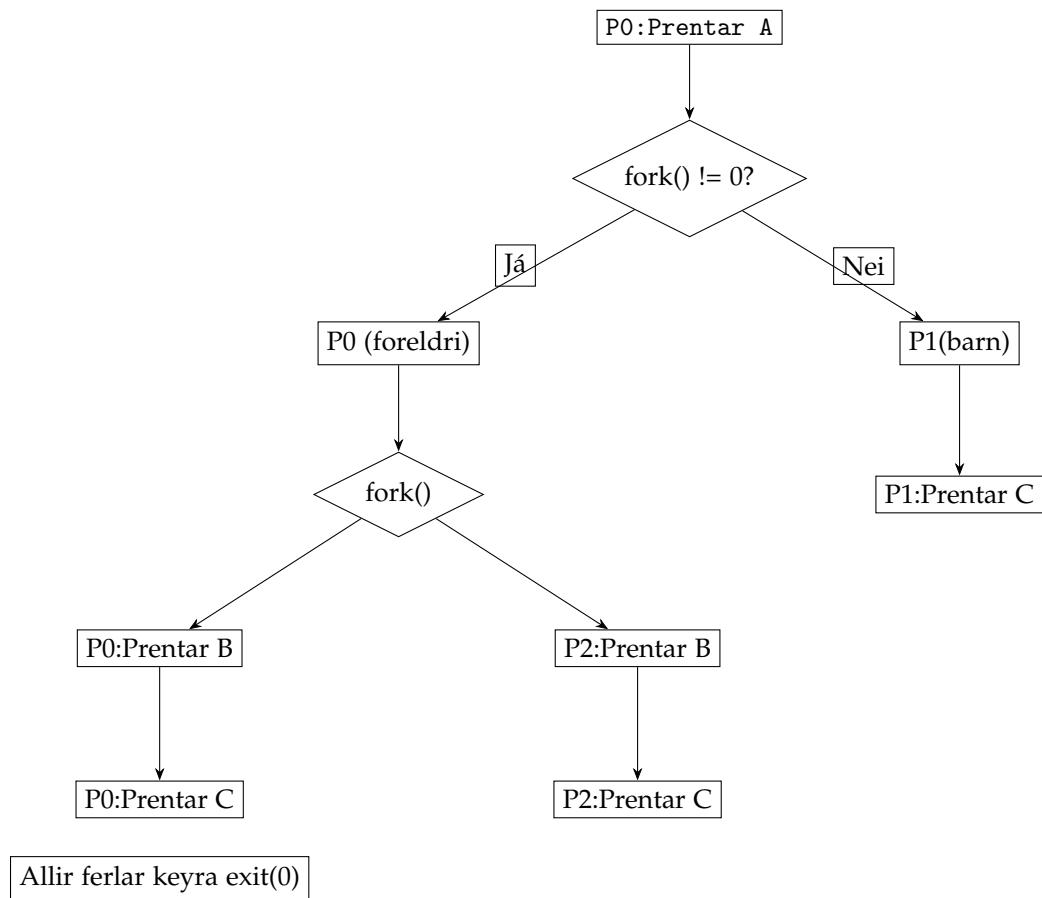
Teiknið upp ferlarit (process graph) fyrir forritið og merkið inná það hvað forritið prentar út á hverjum stað.

Svar:

Skoðum þetta forrit:

1. lína 1: prentar út A
2. lína 2: fallið fall er kallað
3. lína 1 í falli: hér er kallað á fork (ferill P1) og ef það er ekki 0 þá er kallað á fork aftur hér heldur upprunalega ferlið P0 áfram þar sem 1d hjá því er ekki 0 og því er kallað á fork aftur, það myndar nýtt ferli P2 Ferli.
4. lína 3 í falli: Hér prenta ferli P0 og P2 út B en ekki P1 þar sem það stóðst ekki íf skilyrðið þar sem fork() skilar 0.
5. lína 4: prentar út C hér prenta allir 3 ferlarnir út C
6. lína 5: hér er kallað á exit(0) í öllum ferlunum og því er forritið lokið.

Skulum nú gera ferlarit:



b.

Sýnið þrjár ólíkar mögulegar útprentanir sem forritið getur gert.

Svar:

Þar sem allir ferlarnir keyra á sama tíma en bara eftir að er kallað á þá þannig mögulegar útkomur eru:

- A C B C B C
- A B B C C C
- A B C B C C

c.

Hversu mörg ferli munu framkvæma skipunina `exit(0)` í main-fallinu? Rökstyðjið svarið!

Svar:

Allir ferlarnir munu framkvæma skipunina `exit(0)` þar sem það er kallað á hana í main fallinu og því munu allir ferlarnir loka.

Próf 2021 - Og mín Heiðarleg tilraun á að leysa það

1.

Í þessu dæmi höfum við 6-bitu orðið **010011**. Það er hægt að túlka það á þrjá vegu sem gildi: *i.* 6-bitu heiltala án formerkis (unsigned), *ii.* 6-bitu tvíandhverfu heiltala (signed) og *iii.* 6-bitu fleytitala (floating point) með 1 formerkisbita, 3 veldisbita og 2 brotbita.

a.

Túlkið orðið 010011 sem gildi á þessa þrjá vegu. Sýnið útreikning (sérstaklega í fleytitölunni).

Svar:

- **6-bitu heiltala án formerkis:** reiknum þá $2^0 + 2^1 + 2^4 = 19$
- **6-bitu tvíandhverfu heiltala:** Reiknum þá sem signed þar sem það er 0 fremst þá er þetta jákvæð tala og skilar sama svari og án formerkis þannig $2^0 + 2^1 + 2^4 = 19$
- **6-bitu fleytitala:** sem 6 bita fleyti tala með 1 formerkisbita og 3 veldisbita og 2 brotbita þá höfum við 0 fyrir jákvæða tölu 100 sem veldisbita og 11 sem brotbita. við þurfum fyrst að reikna raunverulegan veldisvísir (E) fyrir það þurfum við fyrst að reikna skekkjustuðulinn sem er reiknaður :

$$\text{Bias} = 2^{k-1} - 1 = 2^{3(\text{bitar})-1} - 1 = 2^2 - 1 = 4 - 1 = 3$$

Reiknum nú

$$E = \text{bitagildið þ.e.a.s } 100 = 4 - \text{Bias} = 4 - 1$$

Svo raunverulegi veldisvísirinn er 1 semsagt 2^1 nú finnum við brothlutann 11 við vitum að það er gefinn 1 á undan brothluta svo við fáum 1.11 til að reikna brothluta gerum við $1 \cdot (2^{1/2} + 2^{1/4}) = 1.75$ Setjum þetta nú saman

$$+1.75 \times 2^1 = 3.5$$

Vika 1

Kynning, Linux, C

Vika 2

C, Bendar, minni, notkun

Vika 3

Upplýsingar sem bitar, heiltölur

Vika 4

Bætaröð, fleytitölur

Vika 5

Skipulag örgjava, smalamálsforritun

Vika6

Stýriskipanir og stef í smalamáli

Vika 7

Gögn og yfirflæði minnis

Vika 8

Bestun smalamálskóða

Vika 9

Minnisstigveldi, skyndiminni

Vika 10

Tenging, keyrsluskrár, forritasöfn

Vika 11

Frábrigði, ferlastýring

Vika 12

Sýndarminni

Vika 13

Minnisúthlutun, ruslasöfnun, minnisvillur

Vika 14

Samantekt