Sýndarminni

- sýndarvistföng: a bitar
- raunvistföng: b bitar
-

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 ($i = 0, \ldots 63$) 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:

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

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:

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

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:

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

2

Við höfum 10-bita 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 bita fleytitölur og við höfum að brothlutinn er 5 bitar þá er veldishlutinn 4 bitar og formerkisbitinn er 1 biti

1. Reiknum bias (skekkjustuðulinn)

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

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 <u>stærsta staðlaða talan</u> 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 <u>stærsta staðlaða talan</u> 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:

$$1.11111111_2 \times 2^1 = 1.(2^{1/2^1} + 1/2^2 + 1/2^3 + 1/2^{\cdots} + 2^8) = 1.9960375 \times 2^1$$

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 mminni 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:

```c
unsigned long fun(unsigned long *ptr, unsigned long s, unsigned long x){
    unsigned long rax = *ptr;
    while(s < x){
        rax = 6 * rax;
        s = s + 1;
    }
    return rax;
}
```

c.

Smalamálskóðinn að ofan er úttak úr þýðandanum **gcc** með bestunarrofann **-Og**. Ef notaður er bestunarrofinn **-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 línum í smalamálskóðanum.

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

```c
int rec(int n, int m){
    if(n >= m){
        m = m + m;
        n = n - 2;
        int result = rec(n, m);
        return result + 1;
    }
```

```
        else{
            return 0;
        }
    }
```

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 réttri röð?
8-15	b (8 bæti)	double b
16	c (1 bæti)	char c
17-23	Padding (7 bæti)	Fylling til að halda réttri röð?
24-31	d (8 bæti)	char *d

Færslan tekur því 32 bæti

c.

Skyndiminni er 2-vítt (*2-way set associative*

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.

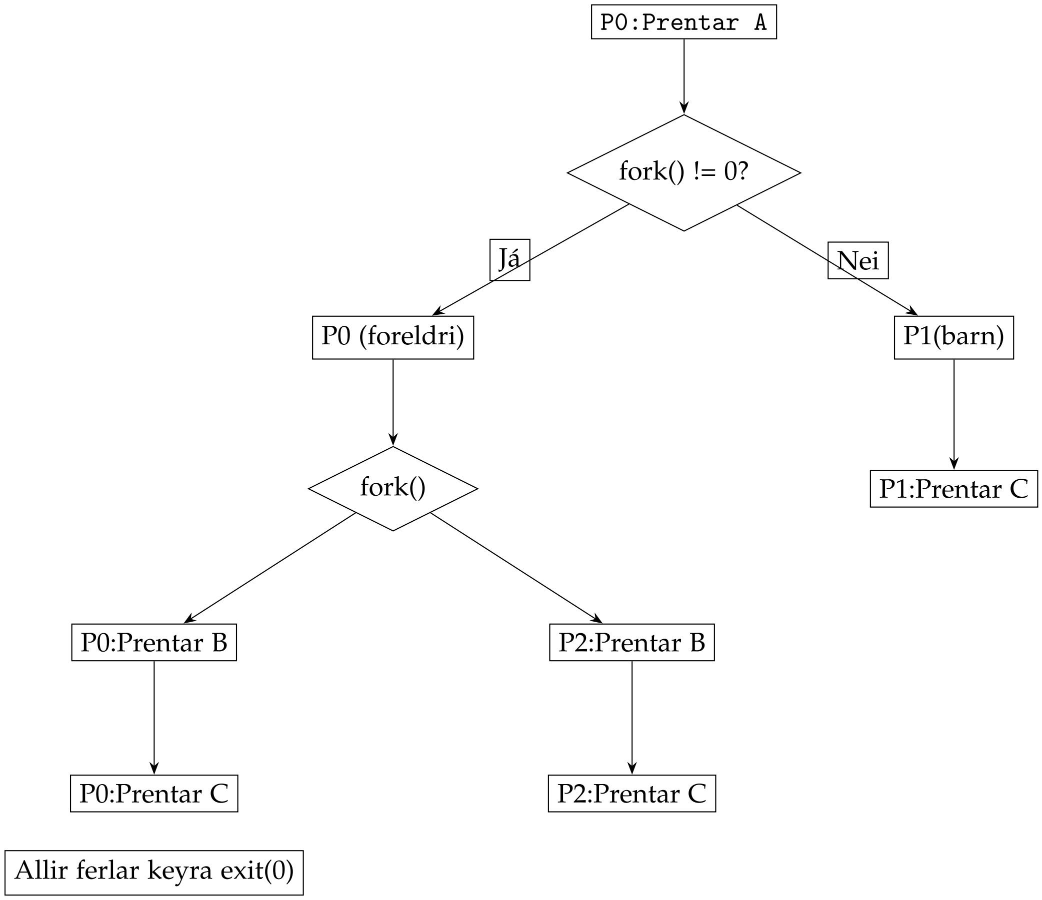
Teiknið upp ferlarit (process graph) fyrir forritið og merkið inná það hvað forritið prentar út á hverjum stað.

Svar:

Skoðum þetta forrit:

1. lína 1: prentar út A
2. lína 2: fallið fall er kallað
3. lina 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 Id 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 if 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 ȩxit(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-bita orðið **010011**. Það er hægt að túlka það á þrjá vegu sem gildi: *i*. 6-bita heiltala án formerkis (unsigned), *ii*. 6-bita tvíandhverfu heiltala (signed) og *iii*. 6-bita 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-bita heiltala án formerkis**: reiknum þá $2^0 + 2^1 + 2^4 = 19$
- **6-bita 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-bita 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 veldisvisi (**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 $1.(2^{1/2} + 2^{1/4}) = 1.75$ Setjum þetta nú saman

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

b.

Þið megið breyta einum bita í orðinu að ofan og viljið fá sem hæst gildi (þ.e. sem næst $+\infty$). Sýnið og rökstyðjið í hverju tilviki hvaða bita þið mynduð breyta til að hámarka gildið. Það geta verið ólíkir bitar í hverri túlkun

svar:

a. í unsigned þá myndum við bæta 1 fremst þar sem hægt veri svo $(1)10011 = 1 + 2 + 16 + 32 = 51$

b. í signed þá má fremsti ekki vera 1 þar sem þá er talan í mínus svo $01(1)011 = 1 + 2 + 8 + 16 = 27$

c. hér myndum við vilja bæta á veldisvísinn svo þá væri talan 011011 Reiknum skekkjuna bias $= 2^{3-1} - 1 = 2^2 - 1 = 4 - 1 = 3$ Reiknum Raunverulegann veldisvísi $e - \text{bias} = 6 - 3 = 3$ Svo talan yrði $1.75 \times 2^3 = 1.75 \times 8 = 14$

c.

Hvaða einum bita ætti að breyta til að lágmarka gildið (þ.e. komast sem næst $-\infty$) í hverri túlkun? Rökstyðjið hvert tilvik

svar:

a. í unsigned þá myndum við taka 0 fremst þar sem hægt veri svo $000011 = 1 + 2 = 3$

b. í signed þá setjum við fremsta sem 1 þar sem þá er talan í mínus svo $(1)10011 = -32 + 16 + 2 + 1 = -13$

c. hér myndum við vilja setja formerkisbitann sem 1 til að fá töluna í mínus svo þá væri talan 110011 Reiknum skekkjuna bias $= 2^{3-1} - 1 = 2^2 - 1 = 4 - 1 = 3$ Reiknum Raunverulegann veldisvísi $e - \text{bias} = 4 - 3 = 1$ Svo talan yrði $-1.75 \times 2^1 = -1.75 \times 2 = -3.5$

2

Hér fyrir neðan er x86-64 smalamálsútgáfa af endurkvæma fallinu func.

```
func:
    cmpl    $1, %edi
    jle     .L3
    pushq   %rbx
    leal    (%rdi,%rdi,2), %ebx
    leal    3(%rdi), %edx
    testl   %edi, %edi
    cmovns  %edi, %edx
    sarl    $2, %edx
    movl    %edx, %edi
    call    func
    addl    %ebx, %eax
    popq    %rbx
    ret
.L3:
    movl    $1, %eax
    ret
```

a.

Skrifið jafngilda C útgáfu af þessu falli. Til að hjálpa ykkur við það er hér fyrir neðan beinagrind af fallinu sem þið getið fyllt inn í. Rökstyðjið sérstaklega hvaða smalamálsskipanir standa á bakvið þann kóða sem þið setjið inn.

```
int func(int n) {
    if ( _____ )
        return _____;
   return _____ + func( _____ );
}
```

svar: Alright skoðum hvað línurnar gera

- Lína 1: berum edi við 1
- Lína 2: ef edi er 1 eða minna þá er hoppað í **.L3**
- Lína 3: ýtt rbx á hlaðann
- Lína 4: rdi sinnum 3 sett í ebx
- Lína 5: 3 + rdi sett í edx
- Lína 6: skoðar hvort edi sé neikvætt eða ekki
- Lína 7: ef edi er ekki neikvætt þá er edx = n annars er edx = n+3
- Lína 8: shift arithmetically right eða $edi >> 2$ sem deilir með 4
- Lína 9: setur edi í edx
- Lína 10: callað á func
- Lína 11: ebx bætt við eax þar sem ebx er 3 * n þannig $eax = eax + ebx$
- Lína 12: rbx poppað af hlaðanum endurheimt gildi úr kösinni
- Lína 13: return skipun
- Lína 14: sett eax sem 1

- lína 15: return skipun

Reynum nú að setja þetta í c fall:

```c
int func(int n){
    if(n >=1)
        return 1;
    return (3*n) + func((n >= 0 ? n : n+3) / 4);
}
```

b.

Teiknið upp hlaðaramma (*stack frame*) fyrir fallið. Sýnið stöðuna þegar þrjú endurkvæm köll hafa orðið. Tilgreinið einstaka hluta hvers hlaðaramma.

3

Hér fyrir neðan er C fall sem afritar eitt stak á milli tveggja tvívíðra fylkja. Fylkin eru víðvær og er fylkið a af stærðinni **MxN**, en **b** er af stærðinni **NxM**. Þið viti ekki gildin á **M** eða **N**, en eigið að geta fundið þau útfrá smalamálskóðanum fyrir fallið sem einnig er gefinn.

```
long int a[M][N];
long int b[N][M];

void afrit( int i, int j ) {
    a[i][j] = b[j][i];
}
```

```
afrit:
    movslq      %edi, %rdi
    movslq      %esi, %rsi
    leaq        0(,%rsi,8), %rax
    subq        %rsi, %rax
    addq        %rdi, %rax
    movq        b(,%rax,8), %rdx
    leaq        (%rdi,%rdi,4), %rax
    leaq        (%rdi,%rax,2), %rax
    addq        %rax, %rsi
    movq        %rdx, a(,%rsi,8)
    ret
```

a.

Hver eru gildin á **M** og **N**? Rökstyðjið svörin út frá skipununum í smalamálskóðanum.

svar: ef við fylgjum diane's silk dress cost 89 þá sjáum við að edi er þar að leiðandi i og esi er þá j.

skoðum nú línurnar

- Lína 1:

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