Ugeopgave 2

Computerarkitektur

Kristian Gausel[1], Rasmus Skovdal[2], og Steffan C. S. Jørgensen[3]

[1]201509079, 201509079@post.au.dk
[2]201509421, rasmus.skovdal@post.au.dk
[3]201505832, 201505832@post.au.dk

1. maj 2016

1 Opgave A

Programmet i kodeudsnit 1 multiplicerer tallene x og y alene ved brug addition og en *while*-løkke.

Kodeudsnit 1: Kildekoden for programmet *imul.c* skrevet i C

```c
#include <stdio.h>

int imul(int x, int y){
  int p;

  p = 0;
  while ( x > 0 ) {
    x = x - 1;
    p = p + y;
  }
  return p;
}

int main(int argc, char *argv[]) {
  printf("imul(%d,%d) = %d.\n",2,3, imul(2,3));
}
```

Funktionen returnerer *0*, når $x \leqslant 0$, da *while*-løkken i så fald aldrig køres mens p instantieres til 0. Når $y < 0$ fungerer funktionen som ventet, da det uden problemer er muligt at addere med et negativt tal.

2 Opgave B

Strukturen af programmet i kodeudsnit 1 findes også i en udgave skrevet i *IJVM*, der kan findes i appendix A. I dette afsnit vil vi behandle programmets stak.

Stakkens maksimale størrelse er 11 under forudsætning af, at $x > 0$. Hvis det derimod er tilfældet, at $x \leqslant 0$, er den maksimale størrelse 10. For $x > 0$ er stakken størst efter, tallet *1* bliver lagt på stakken i linje 19 og senere, når y bliver lagt på i linje 23. Hvis $x \leqslant 0$ vokser stakkens til sin maksimale størrelse på linje 10, hvor tallet *0* lægges på stakken. Stakken når desuden sin maksimale størrelse på linjerne 14, 16 og 28.

3 Opgave C

Efter kommandoen *invokevirtual imul* ser stakken ud som følger:

$$\text{stack} = 15,\ 51,\ 0,\ 3,\ 2,\ 22,\ 0,\ 1,\ 16$$

I tabel 1 ses alle stakkens værdier efter udførslen af kommandoen *invokevirtual imul* og en forklaring af disses betydning.

15	**51**	**0**	**3**	**2**	**22**	**0**	**1**	**16**
Main LV	*Main* PC	p	y	x	Link Pointer	*Initial* LV	*Initial* PC	Link Pointer

Tabel 1: Stakværdier og deres betydning

Det ses altså, at værdien *51* er *main* Program Counter (PC), der peger på den næste instruktion, som skal udføres i *main*-metoden efter udførslen af *invokevirtual*. Denne næste instruktion ligger på plads nummer 51 i programmets *method area*. Hermed kan programmet arbejde videre efter afslutningen af *imul*-metoden.

Værdien *22* er Link Pointer, der peger på *emul*-metoden, og bruges som plads til at opbevare returværdien p af *emul*-metoden. Den har værdien 22, da den peger på *main* PC, der ligger som stakelement nr. 16 (dette ses på den foregående Link Pointer). Da *Main* PC er 6 elementer over *Initial* PC i stakken, peger den nye Link Pointer på stakelement 22.

4 Opgave D

Antallet af operationer udført i programmet *imul.j* kan opskrives som en funktion af x. Programmet består af to dele: *while*-løkken og resten. 13 operationer bruges på hver *while*-løkke, og uagtet værdien af x køres som minimum de to første linjer af *while*-løkken. Dette resulterer i et minimum af 11 operationer, når $x < 0$, og 13 operationer for $x = 0$, da to ekstra operationer udføres i *while-løkken*, når $x = 0$. På baggrund af disse observationer kan vi opskrive udtrykket i ligning 1 for antallet af operationer O som en funktion af værdien for x:

$$O(x) = \begin{cases} 13(x+1) & \text{if } x \geqslant 0 \\ 11 & \text{if } x < 0 \end{cases} \tag{1}$$

5 Opgave E

I dette afsnit vil vi udføre programmet *imul.j* for forskellige værdier af x og antallet af linjer i kørslen stemmer overens med udtrykket fra ligning 1.

$$\begin{aligned} O(0) &= 13(0+1) = 13 \\ O(1) &= 13(1+1) = 26 \\ O(2) &= 13(2+1) = 39 \\ O(-1) &= 11 \end{aligned} \tag{2}$$

På baggrund af de ovenstående forventede resultater og testresultaterne i appendix B kan vi konkludere, at funktionsudtrykket i ligning 1 stemmer overens med vores test.

A Opgave B

Kodeudsnit 2: Kildekoden for programmet *imul.j* skrevet i IJVM

```
1  .method imul
2  .args  3                    // ( int x, int y )
3  .define x = 1
4  .define y = 2
5  .locals 1                   // int p;
6  .define p = 3
7
8          bipush 0
9          istore p            // p = 0;
10
11 while:                      // while
12         iload x
13         iflt end_while
14         iload x
15         ifeq end_while      //    ( x > 0 ) {
16         iload x
17         bipush 1
18         isub
19         istore x            //    x = x - 1;
20         iload p
21         iload y
22         iadd
23         istore p            //    p = p + y;
24         goto while          // }
25 end_while:
26         iload p
27         ireturn             // return p;
28
29 .method main
30 .args  1
31 .define OBJREF = 44
32
33         bipush OBJREF
34         bipush 2
35         bipush 3
36         invokevirtual imul
37         ireturn             // return imul(2,3);
```

B Opgave E

Kodeudsnit 3: Kørsel af *imul.j* for $x = -1$ og $O(-1) = 11$

```
 0                              stack = 0, 1, 16
 1 bipush 44          [10 2c]   stack = 44, 0, 1, 16
 2 bipush -2          [10 fe]   stack = -2, 44, 0, 1, 16
 3 bipush 3           [10 03]   stack = 3, -2, 44, 0, 1, 16
 4 invokevirtual 0    [b6 00 00] stack = 15, 51, 0, 3, -2, 22, 0, 1, 16
 5 bipush 0           [10 00]   stack = 0, 15, 51, 0, 3, -2, 22, 0, 1, 16
 6 istore 3           [36 03]   stack = 15, 51, 0, 3, -2, 22, 0, 1, 16
 7 iload 1            [15 01]   stack = -2, 15, 51, 0, 3, -2, 22, 0, 1, 16
 8 iflt 25            [9b 00 19] stack = 15, 51, 0, 3, -2, 22, 0, 1, 16
 9 iload 3            [15 03]   stack = 0, 15, 51, 0, 3, -2, 22, 0, 1, 16
10 ireturn            [ac]      stack = 0, 0, 1, 16
11 ireturn            [ac]      stack = 0
```

Kodeudsnit 4: Kørsel af *imul.j* for $x = 0$ og $O(0) = 13$

```
 0                              stack = 0, 1, 16
 1 bipush 44          [10 2c]   stack = 44, 0, 1, 16
 2 bipush 0           [10 00]   stack = 0, 44, 0, 1, 16
 3 bipush 0           [10 00]   stack = 0, 0, 44, 0, 1, 16
 4 invokevirtual 0    [b6 00 00] stack = 15, 51, 0, 0, 0, 22, 0, 1, 16
 5 bipush 0           [10 00]   stack = 0, 15, 51, 0, 0, 0, 22, 0, 1, 16
 6 istore 3           [36 03]   stack = 15, 51, 0, 0, 0, 22, 0, 1, 16
 7 iload 1            [15 01]   stack = 0, 15, 51, 0, 0, 0, 22, 0, 1, 16
 8 iflt 25            [9b 00 19] stack = 15, 51, 0, 0, 0, 22, 0, 1, 16
 9 iload 1            [15 01]   stack = 0, 15, 51, 0, 0, 0, 22, 0, 1, 16
10 ifeq 20            [99 00 14] stack = 15, 51, 0, 0, 0, 22, 0, 1, 16
11 iload 3            [15 03]   stack = 0, 15, 51, 0, 0, 0, 22, 0, 1, 16
12 ireturn            [ac]      stack = 0, 0, 1, 16
13 ireturn            [ac]      stack = 0
```

Kodeudsnit 5: Kørsel af *imul.j* for $x = 1$ og $O(1) = 26$

```
 0                              stack = 0, 1, 16
 1 bipush 44          [10 2c]   stack = 44, 0, 1, 16
 2 bipush 1           [10 01]   stack = 1, 44, 0, 1, 16
 3 bipush 1           [10 01]   stack = 1, 1, 44, 0, 1, 16
 4 invokevirtual 0    [b6 00 00] stack = 15, 51, 0, 1, 1, 22, 0, 1, 16
 5 bipush 0           [10 00]   stack = 0, 15, 51, 0, 1, 1, 22, 0, 1, 16
 6 istore 3           [36 03]   stack = 15, 51, 0, 1, 1, 22, 0, 1, 16
 7 iload 1            [15 01]   stack = 1, 15, 51, 0, 1, 1, 22, 0, 1, 16
 8 iflt 25            [9b 00 19] stack = 15, 51, 0, 1, 1, 22, 0, 1, 16
 9 iload 1            [15 01]   stack = 1, 15, 51, 0, 1, 1, 22, 0, 1, 16
10 ifeq 20            [99 00 14] stack = 15, 51, 0, 1, 1, 22, 0, 1, 16
11 iload 1            [15 01]   stack = 1, 15, 51, 0, 1, 1, 22, 0, 1, 16
12 bipush 1           [10 01]   stack = 1, 1, 15, 51, 0, 1, 1, 22, 0, 1, 16
13 isub               [64]      stack = 0, 15, 51, 0, 1, 1, 22, 0, 1, 16
14 istore 1           [36 01]   stack = 15, 51, 0, 1, 0, 22, 0, 1, 16
15 iload 3            [15 03]   stack = 0, 15, 51, 0, 1, 0, 22, 0, 1, 16
16 iload 2            [15 02]   stack = 1, 0, 15, 51, 0, 1, 0, 22, 0, 1, 16
17 iadd               [60]      stack = 1, 15, 51, 0, 1, 0, 22, 0, 1, 16
18 istore 3           [36 03]   stack = 15, 51, 1, 1, 0, 22, 0, 1, 16
19 goto -24           [a7 ff e8] stack = 15, 51, 1, 1, 0, 22, 0, 1, 16
20 iload 1            [15 01]   stack = 0, 15, 51, 1, 1, 0, 22, 0, 1, 16
21 iflt 25            [9b 00 19] stack = 15, 51, 1, 1, 0, 22, 0, 1, 16
22 iload 1            [15 01]   stack = 0, 15, 51, 1, 1, 0, 22, 0, 1, 16
23 ifeq 20            [99 00 14] stack = 15, 51, 1, 1, 0, 22, 0, 1, 16
24 iload 3            [15 03]   stack = 1, 15, 51, 1, 1, 0, 22, 0, 1, 16
25 ireturn            [ac]      stack = 1, 0, 1, 16
26 ireturn            [ac]      stack = 1
```

Kodeudsnit 6: Kørsel af *imul.j* for $x = 2$ og $O(2) = 39$

```
 0                              stack = 0, 1, 16
 1 bipush 44          [10 2c]   stack = 44, 0, 1, 16
 2 bipush 2           [10 02]   stack = 2, 44, 0, 1, 16
 3 bipush 2           [10 02]   stack = 2, 2, 44, 0, 1, 16
 4 invokevirtual 0    [b6 00 00] stack = 15, 51, 0, 2, 2, 22, 0, 1, 16
 5 bipush 0           [10 00]   stack = 0, 15, 51, 0, 2, 2, 22, 0, 1, 16
 6 istore 3           [36 03]   stack = 15, 51, 0, 2, 2, 22, 0, 1, 16
 7 iload 1            [15 01]   stack = 2, 15, 51, 0, 2, 2, 22, 0, 1, 16
 8 iflt 25            [9b 00 19] stack = 15, 51, 0, 2, 2, 22, 0, 1, 16
 9 iload 1            [15 01]   stack = 2, 15, 51, 0, 2, 2, 22, 0, 1, 16
10 ifeq 20            [99 00 14] stack = 15, 51, 0, 2, 2, 22, 0, 1, 16
11 iload 1            [15 01]   stack = 2, 15, 51, 0, 2, 2, 22, 0, 1, 16
12 bipush 1           [10 01]   stack = 1, 2, 15, 51, 0, 2, 2, 22, 0, 1, 16
13 isub               [64]      stack = 1, 15, 51, 0, 2, 2, 22, 0, 1, 16
14 istore 1           [36 01]   stack = 15, 51, 0, 2, 1, 22, 0, 1, 16
15 iload 3            [15 03]   stack = 0, 15, 51, 0, 2, 1, 22, 0, 1, 16
16 iload 2            [15 02]   stack = 2, 0, 15, 51, 0, 2, 1, 22, 0, 1, 16
17 iadd               [60]      stack = 2, 15, 51, 0, 2, 1, 22, 0, 1, 16
18 istore 3           [36 03]   stack = 15, 51, 2, 2, 1, 22, 0, 1, 16
19 goto -24           [a7 ff e8] stack = 15, 51, 2, 2, 1, 22, 0, 1, 16
20 iload 1            [15 01]   stack = 1, 15, 51, 2, 2, 1, 22, 0, 1, 16
21 iflt 25            [9b 00 19] stack = 15, 51, 2, 2, 1, 22, 0, 1, 16
22 iload 1            [15 01]   stack = 1, 15, 51, 2, 2, 1, 22, 0, 1, 16
23 ifeq 20            [99 00 14] stack = 15, 51, 2, 2, 1, 22, 0, 1, 16
24 iload 1            [15 01]   stack = 1, 15, 51, 2, 2, 1, 22, 0, 1, 16
25 bipush 1           [10 01]   stack = 1, 1, 15, 51, 2, 2, 1, 22, 0, 1, 16
26 isub               [64]      stack = 0, 15, 51, 2, 2, 1, 22, 0, 1, 16
27 istore 1           [36 01]   stack = 15, 51, 2, 2, 0, 22, 0, 1, 16
28 iload 3            [15 03]   stack = 2, 15, 51, 2, 2, 0, 22, 0, 1, 16
29 iload 2            [15 02]   stack = 2, 2, 15, 51, 2, 2, 0, 22, 0, 1, 16
30 iadd               [60]      stack = 4, 15, 51, 2, 2, 0, 22, 0, 1, 16
31 istore 3           [36 03]   stack = 15, 51, 4, 2, 0, 22, 0, 1, 16
32 goto -24           [a7 ff e8] stack = 15, 51, 4, 2, 0, 22, 0, 1, 16
33 iload 1            [15 01]   stack = 0, 15, 51, 4, 2, 0, 22, 0, 1, 16
34 iflt 25            [9b 00 19] stack = 15, 51, 4, 2, 0, 22, 0, 1, 16
35 iload 1            [15 01]   stack = 0, 15, 51, 4, 2, 0, 22, 0, 1, 16
36 ifeq 20            [99 00 14] stack = 15, 51, 4, 2, 0, 22, 0, 1, 16
37 iload 3            [15 03]   stack = 4, 15, 51, 4, 2, 0, 22, 0, 1, 16
38 ireturn            [ac]      stack = 4, 0, 1, 16
39 ireturn            [ac]      stack = 4
```