Ugeopgave 4

Computerarkitektur

Kristian Gausel[1], Rasmus Skovdal[2], and 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 Indledning

I denne raport vil vi udvide *Mic1*'s instruktionssæt med tre operationer. Disse er aritmetiske venstre- og højreskift, samt et logisk højreskift. Alle instruktionerne skal tage to argumenter, a og b. Argument a er antallet af skift, der skal udføres, imens argument b er tallet, der skal skiftes.

2 Aritmetisk venstreskift (ishl 0x78)

Som standard understøtter *Mic1*'en kun et aritmetisk venstreskift på 1 byte, dvs. 8 bit, men med instruktionen *ishl*, skal der gøres et 1-bit's venstreskift. Hertil udnyttes, at et 1-bits venstreskift er det samme som at gange med to, hvilket er det samme som at lægge tallet til sig selv.

Kode 1: Instruktionen *ishl*

```
1  ishl = 0x78:
2    MAR = SP = SP - 1; rd     #Read in 2nd element on stack to MDR
3
4      #Make the number 31, used to get the 5 least
5      #  significant bits from the shifting number.
6    OPC = H = 1               #1
7    OPC = H = H + OPC + 1     #3
8    OPC = H = H + OPC + 1     #7
9    OPC = H = H + OPC + 1     #15
10   H   = H + OPC + 1         #31
11   OPC = H AND TOS           #Saving it in OPC for later
12   TOS = H = MDR             #Set TOS and H to the 2nd stack element
13
14     #If the number of shifts is not zero,
15     #  go to loop else end the loop
16   Z = OPC; if(Z) goto ishl_endloop; else goto ishl_loop
17
18 ishl_loop:
19   TOS = H = H + TOS         #Left shift TOS
20
21     #Subtract 1 from OPC and check if it is zero. If it is
22     #  zero (no more shifts) end the loop else goto loop
23   Z = OPC = OPC - 1; if(Z) goto ishl_endloop; else goto ishl_loop
24
25 ishl_endloop:
26   MDR = TOS; wr; goto main #Update MDR to TOS and write and goto main
```

Denne instruktion er afprøvet under forskellige situationer, heriblandt generel brug, negative tal, overløb og for store argumenter.

a	b	Forventet resultat	Opnået resultat
3	3	24	24
1	-1	-2	-2
2	-5	-20	-20
1	$2^{31}-1$	-2	-2
1	-2^{31}	0	0
1	-1696866565	901234166	9012134166
32	1	1	1

Tabel 1: Testresultater for instruktionen *ishl*

3 Aritmetrisk højreskift (ishr 0x7A)

Det aritmetiske højreskift er allerede understøttet i *Mic1*'en, hvorfor instruktionen *ishr* kun skal læse antallet af skift i a og skifte b det specificerede antal gange.

Kode 2: Instruktionen *ishr*

```
1  ishr = 0x7A:
2    MAR = SP = SP - 1; rd     #Read in 2nd element on stack to MDR
3
4      #Make the number 31, used to get the 5 least
5      #  significant bits from the shifting number.
6    OPC = H = 1               #1
7    OPC = H = H + OPC + 1     #3
8    OPC = H = H + OPC + 1     #7
9    OPC = H = H + OPC + 1     #15
10   H   = H + OPC + 1         #31
11   OPC = H AND TOS           #Saving it in OPC for later
12   TOS = MDR
13
14     #If the number of shifts is not zero,
15     #  go to loop else end the loop
16   Z = OPC; if(Z) goto ishr_endloop; else goto ishr_loop
17
18 ishr_loop:
19   TOS = TOS >> 1            #Right shift TOS
20
21     #If there are more shifts to do,
22     #  go to the loop else end the loop
23   Z = OPC = OPC - 1; if(Z) goto ishr_endloop; else goto ishr_loop
24
25 ishr_endloop:
26   MDR = TOS; wr; goto main #Update MDR to TOS and write and goto main
```

Igen har vi afprøvet instruktionen under forskellige situationer. Vi har undersøgt fortegnsbevarelse og for store argumenter.

a	b	Forventet resultat	Opnået resultat
1	-4	-2	-2
2	-17	-5	-5
2	20	2	2
34	20	2	2
15	-1	-1	-1
30	-1073741825	-2	-2

Tabel 2: Testresultater for instruktionen *ishl*

4 Logisk højreskift (iushr 0x7C)

Med instruktionen *ishr* i afsnit 3 bevares fortegn. Instruktionen *iushr* er derimod et logisk højreskift fremfor et aritmetisk højreskift. Altså skal med *iushr* bitmønstret flyttes a gange til højre, hvormed fortegn ændres pga. implemntation af negative tal vha. andenkomplement.

Kode 3: Instruktionen *iushr*

```
1  iushr = 0x7C:
2    MAR = SP = SP - 1; rd     #Read in 2nd element on stack to MDR
3
4      #Make the number 31, used to get the 5 least
5      #  significant bits from the shifting number.
6    OPC = H = 1               #1
7    OPC = H = H + OPC + 1     #3
8    OPC = H = H + OPC + 1     #7
9    OPC = H = H + OPC + 1     #15
10   H   = H + OPC + 1         #31
11   OPC = H AND TOS           #Saving it in OPC for later
12   TOS = MDR                 #Saving the 2nd stack element in TOS
13
14     #Make the number 2^31-1, which is 01111...111 in binary.
15     #  This is used later to remove the signing bit.
16   H = 1 << 8                #2^8
17   H = H >> 1                #2^7
18   H = H << 8                #2^15
19   H = H << 8                #2^23
20   H = H << 8                #2^31 (100...00)
21   H = INV(H)                #!(2^31) = 2^31-1 (011..11)
22
23     #If the number of shifts is not zero,
24     #  go to loop else end the loop
25   Z = OPC; if(Z) goto iushr_endloop; else goto iushr_loop
26
27 iushr_loop:
28   TOS = TOS >> 1            #Right shift TOS
29   TOS = TOS AND H           #Remove the signing bit
30
31     #If there are more shifts to do, go to the loop else end the loop
32   Z = OPC = OPC - 1; if(Z) goto iushr_endloop; else goto iushr_loop
33
34 iushr_endloop:
35   MDR = TOS; wr; goto main #write TOS onto stack, goto main
```

Instruktionen er også blevet afprøvet for brug på positive tal, samt negligering af negative tal for overløb og til sidst en test af hvorvidt negative tal bliver 1 ved maksimal skift.

a	b	Forventet resultat	Opnået resultat
1	8	4	4
2	8	2	2
1	−8	2147483644	2147483644
31	−8	1	1
31	−561	1	1

Tabel 3: Testresultater for instruktionen *iushr*