CompSys 2023

Machine arc
Assembler programming

Johan Topp

4 Machine architecture (about 24 %)

4.1 Assembler programming (about 14 %)

Consider the following program written in RISC-V assembler.

```
<start>:
  li a5,1
  bge a5, a0, L4
  addi a6, a1,4
  slli a0,a0,0x2
  add t1,a1,a0
  li a7,0
  li a0,-1
       L2
  j
   addi a4,a4,1
   slli a4,a4,0x2
  add a4, a1, a4
   sw a2,0(a4)
  addi a7, a7,1
  addi a6, a6, 4
  beq a6, t1, L4
L2:
  lw a2,0(a6)
  mv a4, a7
  mv a5, a6
  bltz a7,L1
L3:
  lw a3,-4(a5)
  bge a2, a3, L1
   sw a3,0(a5)
   addi a4, a4, -1
  addi a5, a5, -4
  bne a4, a0, L3
       L1
L4:
  ret
```

Question 4.1.1: Identify the control structures of the program (e.g. loops, conditionals, and function calls).

Question 4.1.2: Specify which registers are used for pointers. How can you see this? What are the operations?

Question 4.1.3: Which registers contains functions arguments and what are their type? Describe how you identified this.

Question 4.1.4: Rewrite the above X86prime-assembler program to a C program. The resulting program must not have a goto-style and minor syntactical mistakes are acceptable.

Question 4.1.5: Descripe shortly the functionality of the program.

 $\frac{\textbf{Question 4.1.1:}}{\textbf{calls).}} \ \ \textbf{Identify the control structures of the program (e.g. loops, conditionals, and function}$

1	<start>:</start>	
2	li	a5,1
3	bge	a5,a0,L4
4	addi	a6,a1,4
5		a0,a0,0x2
6	add	
7	li	a7,0
8	li	a0,-1
9	j	L2
.0	L1:	
1	addi	a4,a4,1
.2		a4,a4,0x2
.3		a4,a1,a4
4	0.0000000000000000000000000000000000000	a2,0(a4)
.5		a7,a7,1
6		a6,a6,4
.7		a6,t1,L4
.8	L2:	
9	lw	a2,0(a6)
20		a4,a7
21	mv	a5,a6
12		a7,L1
23	L3:	
24	lw	a3,-4(a5)
25	bge	a2,a3,L1
26	sw	a3,0(a5)
27		a4,a4,-1
28		a5,a5,-4
19		a4,a0,L3
30	j	L1
31	L4:	
32	ret	
	•	

```
prolog
      <start>:
2
3
4
5
6
7
8
9
10
11
12
                           check if looop should be run
         bge a5,a0,L4
                           jump til loop
               L2
13
14
15
16
17
         beq a6,t1,L4
18
19
20
21
22
                            restart outer loop
         bltz a7,L1
23
      L3:
24
                            restart outer loop
25
         bge a2,a3,L1
26
27
28
29
                            restart inner loop
         bne a4,a0,L3
30
                            restart outer loop
31
32
                           exit function (jalr x0, x1, 0)
         ret
33
```

Question 4.1.2: Specify which registers are used for pointers. How can you see this? What are the operations?

	10	-							
1	<start>:</start>	1	<start></start>	:		1	<start>:</start>		
2 3 4	li a5,1	2				2	li a5,1		
3	bge a5,a0,L4	3				3	bge a5,a0,L4		
	addi a6,a1,4	4				4	addi a6,a1,4	a6 = a1 + 4 poi	nter? don't know
5	slli a0,a0,0x2	5				5	slli a0,a0,0x2		
6	add t1,a1,a0					6	add t1,a1,a0	t1 = a1 + 4*a0	array offset (assumed
7	li a7,0	7				7	li a7,0		***
8	li a0,-1	8				8	li a0,-1		
9	j L2	9				9	j L2		
.0	L1:	10	L1:			10	L1:		
5 6 7 8 9 10 11 12 14 15 16	addi a4,a4,1	11				11	addi a4,a4,1		
12	slli a4,a4,0x2	12				12	slli a4,a4,0x2		
13	add a4,a1,a4	13				13	add a4,a1,a4	a4 = a1 + 4	pointer
14	sw a2,0(a4)	14	SW	a2,0(a4)	a4 pointer	14	sw a2,0(a4)		•
15	addi a7,a7,1	15				15	addi a7,a7,1		
.6	addi a6,a6,4	16				16	addi a6,a6,4		
.7	beq a6,t1,L4	17				17	beg a6,t1,L4		
.8	L2:	18	L2:			18	L2:		
18 19	lw a2,0(a6)	19	lw	a2,0(a6)	a6 :pointer	19	lw a2,0(a6)	a6 :pointer	
20	mv a4,a7	20			2	20	m∨ a4,a7	A CONTRACTOR OF THE CONTRACTOR	
21	m∨ a5,a6	21				21	mv a5,a6		
22	bltz a7,L1	22				22	bltz a7,L1		
23	L3:	23	L3:			23	L3:		
21 22 23 24 25 26 27	lw a3,-4(a5)	24	lw	a3,-4(a5)	a5 pointer	24	lw a3,-4(a5)	a5 pointer	
25	bge a2,a3,L1	25			•	25	bge a2,a3,L1		
26	sw a3,0(a5)	26	SW	a3,0(a5)	a5 pointer	26	sw a3,0(a5)		
27	addi a4,a4,-1	27				27	addi a4,a4,-1		
28	addi a5,a5,-4	28				28	addi a5,a5,-4		
29	bne a4,a0,L3	29				29	bne a4,a0,L3		
30	j L1	30				30	j L1		
29 30 31	L4:	31	L4:			31	L4:		
32	ret	32	ret			32	ret		
		33				33			
							L.		

Question 4.1.3: Which registers contains functions arguments and what are their type? Describe how you identified this.

```
<start>:
2 3 4 5 6 7 8 9 0 1 1 2 1 3 1 4 1 5 1 6 7
         li
               a5,1
         bge a5, a0, L4
         addi a6,a1,4
         slli a0,a0,0x2
              t1,a1,a0
               a7,0
         li
               a0,-1
               L2
         addi a4,a4,1
         slli a4,a4,0x2
         add a4,a1,a4
              a2,0(a4)
         addi a7, a7, 1
         addi a6,a6,4
         beq a6,t1,L4
8.
      L2:
19
               a2,0(a6)
               a4, a7
1
               a5, a6
22
         bltz a7,L1
      L3:
24
               a3,-4(a5)
25
         bge a2, a3, L1
         sw a3,0(a5)
27
         addi a4,a4,-1
8
         addi a5,a5,-4
29
         bne a4, a0, L3
30
               L1
31
         ret
```

```
<start>:
         li
              a5,1
 3
         bge a5, a0, L4
                           a0 has not been assigned
 4
         addi a6,a1,4
                           al has not been assigned
              a0,a0,0x2
              t1,a1,a0
         li
              a7,0
         li
 8
              a0,-1
 9
              L2
10
      L1:
11
         addi a4,a4,1
12
         slli a4,a4,0x2
13
         add a4, a1, a4
14
              a2,0(a4)
15
         addi a7,a7,1
16
         addi a6, a6, 4
17
         beq
              a6,t1,L4
18
      L2:
19
              a2,0(a6)
20
              a4, a7
21
              a5, a6
22
         bltz a7,L1
23
      L3:
24
              a3,-4(a5)
25
         bge a2, a3, L1
26
              a3,0(a5)
27
         addi a4,a4,-1
28
         addi a5, a5, -4
29
         bne a4, a0, L3
30
              L1
31
      L4:
32
         ret
33
```

Start fra <start> og se hvad der ikke er assigned, før det bliver brugt.

Note:

Husk at følge jumps/branches, ikke bare se oppe fra og ned. L2 kunne benytte et argument der ikke var assigned, men som efterfølgende blev assigned i L1

Question 4.1.4: Rewrite the above X86prime-assembler program to a C program. The resulting program must not have a goto-style and minor syntactical mistakes are acceptable.

```
<start>:
     <start>:
                          2
                                       a5,1
                                  li
        li
             a5,1
                                   bge a5, a0, L4
                                                     if 1>=a0 <=>
                                                                    if a0<1 goto return
        bge a5, a0, L4
                                                     tmp \ a6 = a1[1]
                                   addi a6,a1,4
        addi a6,a1,4
                                  slli a0,a0,0x2
        slli a0,a0,0x2
                                  add t1,a1,a0
                                                     t1 = a1[a0]
                                                                      (length, vides ikke endnu, men antages)
             t1,a1,a0
                                        a7,0
                                                     a7 = 0
             a7,0
                                  li
8
                          8
             a0,-1
                                  li
        li
                                        a0,-1
9
                          9
                                        L2
             L2
.0
                          10
                                L1:
.1
        addi a4,a4,1
                          11
                                   addi a4, a4, 1
2
        slli a4,a4,0x2
                          12
                                   slli a4,a4,0x2
13
        add a4, a1, a4
                          13
                                   add a4, a1, a4
        a2,0(a4)
                          14
                                   a2,0(a4)
.5
        addi a7, a7,1
                          15
                                   addi a7, a7, 1
6
        addi a6, a6, 4
                          16
                                   addi a6, a6, 4
7
                          17
        beq a6,t1,L4
                                   beq a6, t1, L4
8.
                          18
     L2:
                                L2:
9
                          19
                                        a2,0(a6)
                                                    a2 = *a[1] (antager a6 = i i while loopet)
             a2,0(a6)
                                   lw
20
                          20
                                        a4, a7
             a4, a7
                          21
1
                                        a5, a6
                                                    a5 = 1
             a5, a6
                          22
22
                                   bltz a7,L1
                                                    if a7 < 0 goto L1
        bltz a7,L1
23
                          23
     L3:
                               L3:
24
                          24
                                  lw
                                       a3,-4(a5)
                                                    a3 = a[1-1]
        lw
             a3,-4(a5)
                          25
25
                                   bge a2, a3, L1
                                                    if a1[1] >= a1[1-1] goto L1
        bge a2, a3, L1
                          26
                                        a3,0(a5)
                                                    a[1] = a3 \iff a[1] = a[1-1]
26
        sw a3,0(a5)
                          27
                                   addi a4,a4,-1
                                                                     Tyder på at a4 itererer, og dermed er en variabel (j)
27
        addi a4,a4,-1
                          28
                                   addi a5, a5, -4
                                                     a5 = a[1-1]
18
        addi a5, a5, -4
                                                    if -1 != -1 kør indre loop igen., så a4 må være j værdi, da dette er condition variabel
                                   bne a4, a0, L3
29
        bne a4, a0, L3
30
             L1
```

31

ret

Lav konklusioner. Bne I linje 29 er indre loop af j. Og (j--(ud fra linje 27))

Question 4.1.4: Rewrite the above X86prime-assembler program to a C program. The resulting program must not have a goto-style and minor syntactical mistakes are acceptable.

```
<start>:
                               <start>:
                                  li a5,1
        li a5,1
3 4 5
        bge a5, a0, L4
                                  bge a5, a0, L4
                                                   if 1>=a0 <=> if a0<1 goto return
        addi a6,a1,4
                                  addi a6,a1,4
                                                   tmp \ a6 = a1[1]
                                  slli a0,a0,0x2
        slli a0,a0,0x2
                                  add t1,a1,a0
                                                   t1 = a1[a0]
                                                                   (length, vides ikke endnu, men antages)
        add t1,a1,a0
7
                                  li
                                       a7,0
                                                   a7 =0
             a7,0
                                  li
                                       a0,-1
                                                   a0 -=1
        li
             a0,-1
                          9
9
                                  j
                                       L2
             L2
                         10
                              L1:
0
     L1:
                         11
1
                                  addi a4, a4, 1
        addi a4,a4,1
                         12
                                  slli a4,a4,0x2
.2
        slli a4,a4,0x2
                         13
                                  add a4, a1, a4
13
        add a4, a1, a4
                         14
                                  sw a2,0(a4)
4
        sw = a2,0(a4)
                         15
                                  addi a7, a7, 1
.5
        addi a7,a7,1
                         16
                                  addi a6, a6, 4
6
        addi a6, a6, 4
                         17
                                  beq a6, t1, L4
7
        beq a6,t1,L4
                         18
                              L2:
8.
     L2:
                         19
                                  lw
                                       a2,0(a6)
                                                   a2 = *a[i] (antager a6 = i i while loopet)
.9
        lw a2,0(a6)
                         20
                                  mv
                                       a4, a7
                                                   a4 = j
20
             a4,a7
                         21
                                       a5, a6
                                                   a5 = i
                                 mv
11
             a5, a6
                         22
                                 bltz a7,L1
                                                   if a7 < 0 goto L1
22
        bltz a7,L1
                              L3:
                         23
23
     L3:
                         24
                                                   a3 = a[i-1]
                                  lw
                                       a3,-4(a5)
24
        lw a3,-4(a5)
                         25
                                  bge a2, a3, L1
                                                   if a1[i] >= a1[j] goto L1
25
        bge a2,a3,L1
                         26
                                       a3,0(a5)
                                                   a[j] = a3 <=> a[j]=a[j-1]
26
        sw a3,0(a5)
                         27
                                  addi a4,a4,-1
                                                                   Tyder på at a4 itererer, og dermed er en variabel (j)
                                                   a4 -= 1
27
        addi a4,a4,-1
                         28
                                  addi a5, a5, -4
                                                   a5 = a[j-1]
18
        addi a5, a5, -4
                         29
                                  bne a4,a0,L3
                                                   if -1 != -1 kør indre loop igen., så a4 må være j værdi, da dette er condition variabel
29
        bne a4, a0, L3
                         30
                                       L1
30
             L1
                         31
                               L4:
31
                         32
                                  ret
                         33
        ret
```

Question 4.1.4: Rewrite the above X86prime-assembler program to a C program. The resulting program must not have a goto-style and minor syntactical mistakes are acceptable.

```
<start>:
                                <start>:
                                                   <start>:
        li a5,1
                                   li a5,1
                                                      li a5.1
        bge a5, a0, L4
                                   bge a5, a0, L4
                                                      bge a5, a0, L4
                                                                       if 1>=a0 <=> if a0<1 goto return
        addi a6,a1,4
                                   addi a6, a1,4
                                                      addi a6,a1,4
                                                                       tmp a6 = a1[i] (antager a6 = i*4, svarende til "i" i while loopet)
5
                                   slli a0,a0,0x2
        slli a0,a0,0x2
                                                      slli a0,a0,0x2
                                                                       a0 = length *4
                                   add t1,a1,a0
                                                      add t1,a1,a0
                                                                       t1 = &a1[a0]
                                                                                           (length, vides ikke endnu, men antages)
            t1,a1,a0
                                   li
                                        a7,0
                                                      li
                                                          a7,0
                                                                       a7 =j
        li
             a7,0
8
                           8
                                   li
                                        a0,-1
                                                      li
                                                           a0,-1
                                                                       a0 = -1
             a0,-1
        li
                           9
                                                           L2
9
                                   j
                                        L2
              L2
                          10
                                L1:
                                                   L1:
0
     L1:
                          11
.1
                                   addi a4, a4, 1
                                                      addi a4,a4,1
                                                                        j++
        addi a4,a4,1
                          12
                                   slli a4,a4,0x2
                                                      slli a4, a4, 0x2
                                                                        j*4
2
        slli a4,a4,0x2
                          13
                                   add a4,a1,a4
                                                      add a4, a1, a4
                                                                       udregna[j+1]
13
        add a4,a1,a4
                          14
                                   sw a2,0(a4)
                                                      sw a2,0(a4)
                                                                        a1[j+1] = tmp
4
        a2,0(a4)
                          15
                                   addi a7,a7,1
                                                      addi a7, a7,1
                                                                        j reset ud fra at vi så at a7=j i linje 20
.5
        addi a7, a7,1
                          16
                                   addi a6, a6, 4
                                                      addi a6, a6, 4
                                                                        a[i++]
6
        addi a6, a6, 4
                          17
                                   beq a6, t1, L4
                                                      beq a6,t1,L4
                                                                       if &a1[i] = &a[a0] return
7
        beq a6,t1,L4
                          18
                                L2:
                                                   L2:
8.
     L2:
                          19
                                        a2,0(a6)
                                   lw
                                                      lw a2,0(a6)
                                                                      a2 = a[i] evt kald a2 tmp
9
             a2,0(a6)
                          20
                                        a4, a7
                                   mv
                                                      mv a4,a7
                                                                      a4 = j
                                                                                 \langle = \rangle j = i-1
20
             a4, a7
                          21
                                   mv
                                        a5, a6
                                                      mv a5, a6
                                                                      a5 = i
11
        mv
             a5, a6
                          22
                                   bltz a7,L1
                                                      bltz a7,L1
                                                                      if a7 < 0 goto L1
22
        bltz a7,L1
                          23
                                L3:
                                                   L3:
23
     L3:
                          24
                                        a3,-4(a5)
                                                      lw
                                                           a3,-4(a5)
                                                                        a3 = a[j] = a[i-1] i første runde
14
        lw a3,-4(a5)
                          25
                                   bge a2, a3, L1
                                                      bge a2, a3, L1
                                                                        if tmp >= a1[j] goto L1
25
        bge a2, a3, L1
                          26
                                        a3,0(a5)
                                                           a3,0(a5)
                                                                        a[j] = a3 <=> a[j]=a[j-1]
26
        sw a3,0(a5)
                          27
                                   addi a4,a4,-1
                                                      addi a4,a4,-1
                                                                        a4 -= 1
                                                                                         Tyder på at a4 itererer, og dermed er en variabel (j)
27
        addi a4,a4,-1
                          28
                                   addi a5, a5, -4
                                                      addi a5, a5, -4
                                                                        a5 = a[j-1]
18
        addi a5, a5, -4
                          29
                                   bne a4,a0,L3
                                                      bne a4, a0, L3
                                                                        if j != -1 kør indre loop igen., så a4 må være j værdi, da dette er condition
29
        bne a4, a0, L3
                          30
                                        L1
                                                                                                                                                   variabel
30
              L1
                          31
                                L4:
                                                   L4:
31
                          32
                                   ret
                                                      ret
                          33
        ret
```

Question 4.1.4: Rewrite the above X86prime-assembler program to a C program. The resulting program must not have a goto-style and minor syntactical mistakes are acceptable.

```
<start>:
2 3 4 5 6 7 8 9 0 1 1 2 1 3 1 4 1 5 1 6 7
         li a5,1
         bge a5,a0,L4
         addi a6,a1,4
         slli a0,a0,0x2
              t1,a1,a0
               a7,0
         li
               a0,-1
               L2
         addi a4,a4,1
         slli a4,a4,0x2
         add a4, a1, a4
               a2,0(a4)
         addi a7, a7, 1
         addi a6, a6, 4
         beq a6,t1,L4
8.
19
               a2,0(a6)
               a4, a7
1
               a5,a6
22
         bltz a7,L1
24
               a3,-4(a5)
25
         bge a2, a3, L1
               a3,0(a5)
         addi a4,a4,-1
8
         addi a5, a5, -4
29
         bne a4, a0, L3
30
         ret
```

Tid til at oversætte til C kode:

```
void sort(int length, int array[]) {
    int i = 1;
    int lasteIm = array + length;
        while (&array[i] < lasteIm) {
            int j = i-1;
            tmp = array[i];
            while (j>=0 && array[j]> tmp) {
                 array[j+1] = array[j];
                 j--;
            }
            array[j+1] = tmp;
        }
}
```

```
void sort(int length, int array[]) {
C kode:
                                                                       int i = 1;
void sort(int length, int array[]) {
                                                                       if (length >= i)
          int i = 1;
                                                                                  int lastelm = array + length;
          int lastelm = array + length;
                                                                                  int jbase = 0;
          while (&array[i] < lastelm) {
                                                                                  int j = jbase;
                     int j = i-1;
                                                                                  while (&array[i] < lastelm) {
                     tmp = array[i];
                                                                                             int jbase ++;
                     while (j>=0 && array[j]> tmp) {
                                                                                             j = jbase;
                                array[j+1] = array[j];
                                                                                             tmp = array[i];
                                j--;
                                                                                             while (j>=0 && array[j]> tmp) {
                                                                                                       array[j+1] = array[j];
                     array[j+1] = tmp;
                                                                                                       j--;
           }
                                                                                             array[j+1] = tmp;
```

Question 4.1.5: Descripe shortly the functionality of the program.

Funktionen er insertionsort.

Den leverede tabel sorteres gradvist. Den ydre løkke gennemløber alle elementerne et for et og udvider den sorterede del af tabellen. Den indre løkke placerer hvert element det rigtige sted blandt de sorterede elementer.