

link til RISC-V Simulator :

<https://www.cs.cornell.edu/courses/cs3410/2019sp/riscv/interpreter/#>

Nedenfor følger en kort gennemgang af nogle af de vigtigste RISC-V instruktioner fra et "høj-niveau-sprogs-perspektiv". Til sidst er der selvfølgelig nogle opgaver 😊

Aritmetiske operationer

gemmer tallet 10 i en variabel (register t0)	$t0 = 20$	addi t0, x0, 10
gemmer tallet 20 i en variabel (register t1)	$t1 = 10$	addi t1, x0, 20
gemmer summen af de to variabler (registrer) i en ny variabel (nyt register)	$t2 = t0 + t1$	add t2, t0, t1
gemmer differensen af de to variabler	$t3 = t0 - t1$	sub t3, t0, t1

if-statements lavet med betinget branching

I RISC-V er der ikke indbygget if-statements i assembly, men til gengæld har man muligheden for at hoppe til nye linjer i instruktionerne baseret på forskellige betingelser...

hop til "flag1" hvis $t0 \neq t1$	findes ikke længere i højniveau-sprog	bne t0, t1, flag1 flag1: ... "..." viser at der kunne stå flere instruktioner her
hop til "flag1" hvis $t0 = t1$	-	beq t0, t1, flag1 flag1: ... "..." viser at der kunne stå flere instruktioner her
hop til flag hvis $t0 \leq 0$	-	bge t0, x0, flag1 flag1: ... "..." viser at der kunne stå flere instruktioner her

Eksempler på forskellige if-statements og tilsvarende assembly-programmer

flowchart	høj-niveau	assembly	
<pre> graph TD A[t=10] --> B{t=0?} B -- ja --> C[t=20] B -- nej --> D[t = t + 40] C --> D </pre>	<pre> t0 = 10 if(t0 == 0){ t0 = 20; } t0 += 40 </pre>	<pre> addi t0, x0, 10 bne t0, x0, hop addi t0, x0, 20 hop: addi t0, t0, 40 </pre>	Hvilken værdi får t0?
<pre> graph TD A[t=10] --> B{t!=0?} B -- ja --> C[t=20] B -- nej --> D[t = t + 40] C --> D </pre>	<pre> t0 = 10 if(t0 != 0){ t0 = 20; } t0 += 40 </pre>	<pre> addi t0, x0, 10 beq t0, x0, hop1 addi t0, x0, 20 hop1: addi t0, t0, 40 </pre>	Hvilken værdi får t0?
<pre> graph TD A[t=10] --> B{t < 0?} B -- ja --> C[t=20] B -- nej --> D[t = t + 40] C --> D </pre>	<pre> t0 = 10 if(0 > t0){ t0 = 20; } t0 += 40 </pre>	<pre> addi t0, x0, 10 bge t0, x0, hop addi t0, x0, 20 hop: addi t0, t0, 40 </pre>	Hvilken værdi får t0?

For-loops skabt ved hjælp af betinget branching og jumps

jump til label "flag"	jal x0, flag ... flag: <i>"..." viser at der kunne stå flere instruktioner her</i>
-----------------------	---

Eksempel på for-loop og et tilsvarende assembly program

flowchart	højniveau	assembly
<pre> graph TD A[t3 = 0] --> B[t0 = 0] B --> C{t0 < 5} C -- ja --> D[t3 += 10] D --> E[t0 += 1] E --> C C -- nej --> F([stop]) </pre>	<pre> int t3 = 0; for(int t0=0; t0<5;t0++){ t3 +=10; } </pre>	<pre> addi t3, x0, 0 addi t1, x0, 5 #for-loop "5" addi t0, x0, 0 #for-loop variabel "t0" loop_start: #for-loop betingelse forsæt når t0<5 bge t0, t1, loop_end addi t3, t3, 10 #inde i for-loop #for-loop inkrementer "t0" med 1 addi t0, t0, 1 #for-loop gentag jal x0, loop_start loop_end: </pre>

Eksempel på funktioner eller subrutine - "tilsvarende" implementeret i assembly:

Funktionen "setup" sætter variablen t0 = 10 og kalder funktionen "fun" der sætter t0 = 20	<pre>void setup(){ t0 = 10; fun(); } void fun(){ t0 = 20; }</pre>	<pre>setup: addi t0, x0, 10 jal t1, fun jal x0, end fun: addi t0, x0, 20 jalr x0, t1, 0 end:</pre>
---	--	--

Load og store in memory:

Hukommelses-adresserne er :	0 , 4, 8 , 12 osv.
------------------------------------	--------------------

Gemmer i hukommelsen:	
Indlæsning af værdien 117 til adresse 0	<pre>addi x5, x0, 117 sw x5, 0(x0)</pre>
Indlæsning af værdien 211 til adresse 4	<pre>addi x5, x0, 211 sw x5, 4(x0)</pre>

Læser fra hukommelsen:	
Indlæser fra plads "0" til register "x5"	<pre>lw x5, 0(x1)</pre>
Vælger den næste plads i hukommelsen og indlæser til register "x5"	<pre>addi x1, x1, 4 lw x5, 0(x1)</pre>

Division ved hjælp af shift:

Når man dividerer med 10 i ti-talsystemet er det ret nemt - resultatet er ganske enkelt
dividenden rykket én mod højre:

Eksempel :

$120 / 10 = 12$ altså vi fjerner nullet

I to-talsystemet fungerer tilsvarende hvis man dividerer med 2..

Eksempel:

$110 / 10 = 11$ vi fjerner nullet

110 svarer til 6 i ti-talsystemet, 10 svarer til 2 i ti-talsystemet, 11 svarer til 3 i ti-talsystemet

Gemmer tallet 6 i t0	addi t0, x0, 6
Shift'er t0 1 gang med højre og gemmer i t1 (svarer til at dividere med 2) t0 indeholder 6 t1 indeholder 3	srai t1, t0, 1

Opgaver:

1. Skriv assembly program, der simulerer følgende høj-niveau kode:

```
int a = 20;  
int b = 30;  
int c = a + b;  
int d = b - a;  
int e = d/4;
```

2. Skriv assembly program, der simulerer følgende høj niveau kode:

```
int i=50;  
while(i>0){  
    i = i - 10;  
}
```

3. Læs 8 heltals-array ind i hukommelsen

```
int[ ] list = { 8, 16, 24, 32, 40, 48, 56, 64}
```

4. Lav en subrutine til at beregne summen af arrayet fra hukommelsen
5. Lav en subrutine til at beregne gennemsnittet arrayet fra hukommelsen (den skal anvende subrutinen fra før)