SimCPU15 Assembler

SimCPU15 Assembler

Machine instructions are what the SimCPU15 processor understands
(Shown: Prime test)

```
01: 0xf0000000  17: 0x7e000013
02: 0xb4001000  18: 0xb000004e
03: 0xd1000000  19: 0x60000014
04: 0xb4000001  20: 0xb0000059
05: 0xa1000011  21: 0xe0000001
06: 0xbc000002  22: 0xb400000a
07: 0xb4001000  23: 0xe4000001
08: 0xc9000000  24: 0x00000000
09: 0x5b000000
10: 0xb4000000
11: 0x7900000e
12: 0xb4000001
13: 0x1d000000
14: 0x60000006
15: 0xb4001000
16: 0xc9000000
```

SimCPU15 Assembler

Machine instructions are what the SimCPU15 processor understands
(Shown: Prime test)

```
01: 0xf0000000
02: 0xb4001000
03: 0xd1000000
04: 0xb4000001
05: 0xa1000011
06: 0xbc000002
07: 0xb4001000
08: 0xc9000000
09: 0x5b000000
10: 0xb4000000
11: 0x7900000e
12: 0xb4000001
13: 0x1d000000
14: 0x60000006
15: 0xb4001000
16: 0xc9000000
17: 0x7e000013
18: 0xb000004e
19: 0x60000014
20: 0xb0000059
21: 0xe0000001
22: 0xb400000a
23: 0xe4000001
24: 0x00000000
```

Cumbersome to enter programs like this.

SimCPU15 Assembler

Prime test program using assembler instructions:

```
01: in  r0, 0       17: jeq r3, r2, 19
02: mov r1, 4096    18: mov r0, 78
03: st  r0, r1      19: jmp 20
04: mov r1, 1       20: mov r0, 89
05: jle r0, r1, 17  21: out r0, 1
06: mov r3, 2       22: mov r1, 10
07: mov r1, 4096    23: out r1, 1
08: ld  r2, r1      24: hlt
09: mod r2, r3
10: mov r1, 0
11: jeq r2, r1, 14
12: mov r1, 1
13: add r3, r1
14: jmp 6
15: mov r1, 4096
16: ld  r2, r1
```

SimCPU15 Assembler

Prime test program using assembler instructions:

```
01: in  r0, 0       17: jeq r3, r2, 19
02: mov r1, 4096    18: mov r0, 78
03: st  r0, r1      19: jmp 20
04: mov r1, 1       20: mov r0, 89
05: jle r0, r1, 17  21: out r0, 1
06: mov r3, 2       22: mov r1, 10
07: mov r1, 4096    23: out r1, 1
08: ld  r2, r1      24: hlt
09: mod r2, r3
10: mov r1, 0
11: jeq r2, r1, 14
12: mov r1, 1
13: add r3, r1
14: jmp 6
15: mov r1, 4096
16: ld  r2, r1
```

- Assembler: A tool that translates assembler instructions to machine instructions
- Easier to read and write as no decoding has to be done

- We provide an assembler as tool to write your own programs (optional)
- Our assembler (or assemblers in general) can do more than what you see on the left side

SimCPU15 Assembler

Prime test program using assembler instructions:

```
01: in  r0, 0       17: jeq r3, r2, 19
02: mov r1, 4096    18: mov r0, 78
03: st  r0, r1      19: jmp 20
04: mov r1, 1       20: mov r0, 89
05: jle r0, r1, 17  21: out r0, 1
06: mov r3, 2       22: mov r1, 10
07: mov r1, 4096    23: out r1, 1
08: ld  r2, r1      24: hlt
09: mod r2, r3
10: mov r1, 0
11: jeq r2, r1, 14
12: mov r1, 1
13: add r3, r1
14: jmp 6
15: mov r1, 4096
16: ld  r2, r1
```

What would happen if an additional instruction is inserted before line 06?

SimCPU15 Assembler

Prime test program using assembler instructions:

```
01:  in   r0, 0
02:  mov  r1, 4096
03:  st   r0, r1
04:  mov  r1, 1
05:  jle  r0, r1, 17
06:  mov  r3, 2
07:  mov  r1, 4096
08:  ld   r2, r1
09:  mod  r2, r3
10:  mov  r1, 0
11:  jeq  r2, r1, 14
12:  mov  r1, 1
13:  add  r3, r1
14:  jmp  6
15:  mov  r1, 4096
16:  ld   r2, r1
17:  jeq  r3, r2, 19
18:  mov  r0, 78
19:  jmp  20
20:  mov  r0, 89
21:  out  r0, 1
22:  mov  r1, 10
23:  out  r1, 1
24:  hlt
```

What would happen if an additional instruction is inserted before line 06?

Most jump targets would have to be adapted by hand!

SimCPU15 Assembler

Introducing labels:

```
01:          in  r0, 0
02:          mov r1, 4096
03:          st  r0, r1
04:          mov r1, 1
05:          jle r0, r1, noprim
06:          mov r3, 2
07: loop:    mov r1, 4096
08:          ld  r2, r1
09:          mod r2, r3
10:          mov r1, 0
11:          jeq r2, r1, output
12:          mov r1, 1
13:          add r3, r1
14:          jmp loop
15: output:  mov r1, 4096
16:          ld  r2, r1
17:          jeq r3, r2, prim
18: noprim:  mov r0, 78
19:          jmp end
20: prim:    mov r0, 89
21:          out r0, 1
22: end:     mov r1, 10
23:          out r1, 1
24:          hlt
```

- Assign symbols to instructions
- Use symbols as jump targets
- Assembler translates labels into effective position of instruction
- Makes programs easier to read and adapt

SimCPU15 Assembler

More assembler features:

```
<snipsnap>
14:  jmp loop
15:  mov r1, 0x1000
16:  ld  r2, r1
17:  jeq r3, r2, prim     ; if n == d goto prime
18:  mov r0, 'N'
19:  jmp end
20:  mov r0, 'Y'
21:  out r0, 1            ; output result
22:  mov r1, 10           ; newline
23:  out r1, 1
24:  hlt
```

hexadecimal constants (prefix `0x`)

line comments: similar as // in C++, but using semicolon ;

character constants

SimCPU15 Assembler

Usage in a terminal (assume you have compiled simcpu15_asm already):

- `./simcpu15_asm < input.asm > output.bin`
- E.g. for prime.asm:
  `./simcpu15_asm < prime.asm > prime.bin`