An Introduction to the SM213 Simple Machine

Adapted, with permission from materials developed by:

Dr. Mike Feely, UBC

C

```
int a;    // an int
int b[10];// array of 10 ints

void foo () {
  a    = 2;
  b[a] = a;
}
```

ASM

```
ld $2, r0
ld $0x1000, r1
st r0, (r1)

ld (r1), r2
ld $0x2000,  r3
st r2,  (r3,r2,4)
```

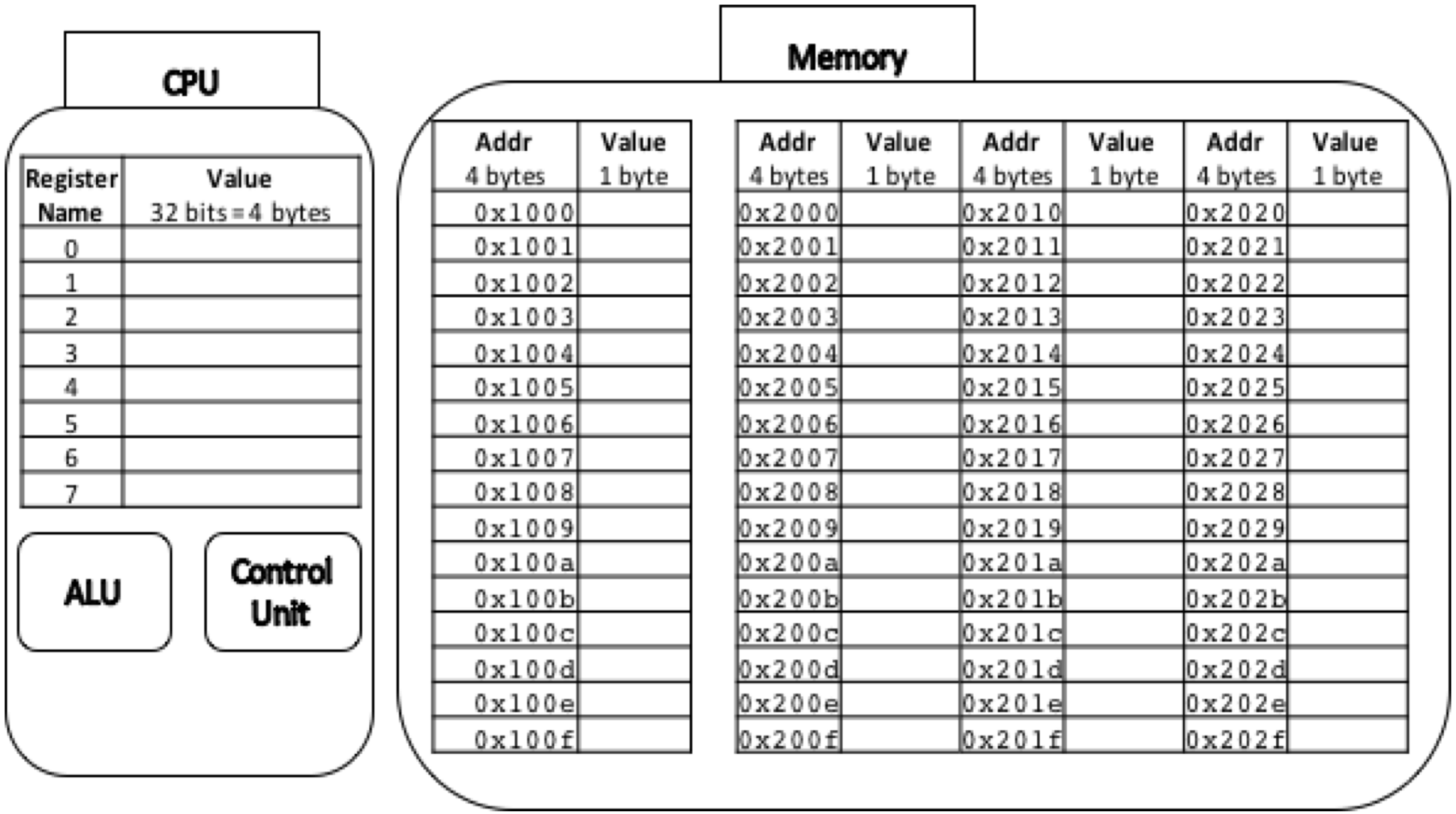
Machine Code

```
00-- 0x2
01-- 0x1000
3001

1012
03-- 0x2000
4232
```

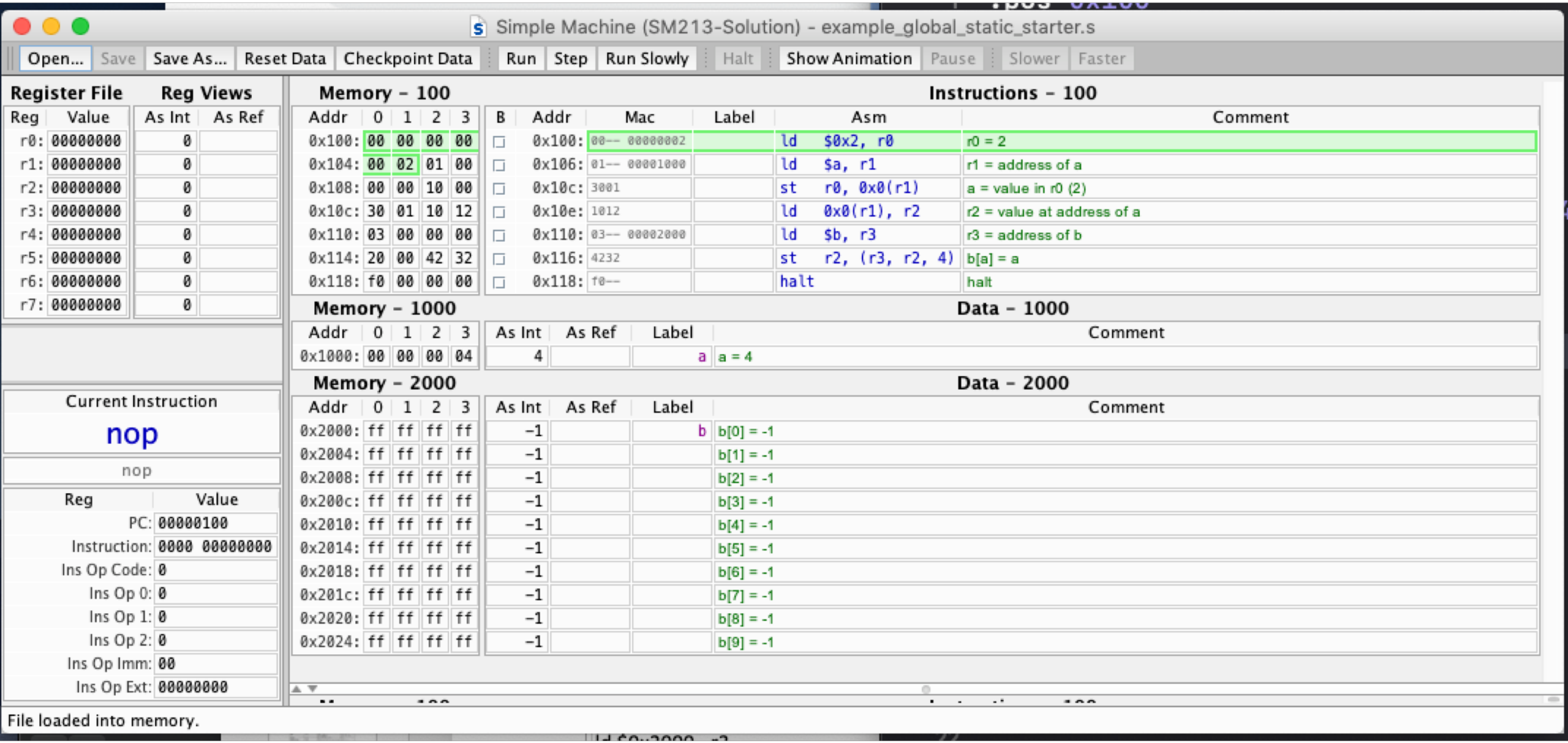
Simple Machine .s file

```
.pos 0x100
          ld   $2,   r0          # r0 = 2
          ld   $a,   r1          # r1 = address of a
          st   r0,  (r1)         # a = value in r0 (2)
          ld   (r1), r2          # r2 = value at address of a
          ld   $b, r3            # r3 = address of b
          st   r2, (r3, r2, 4)   # b[a] = a
          halt                   # halt

.pos 0x1000
a:        .long 0x4               # a = 4
.pos 0x2000
b:        .long 0xffffffff        # b[0] = -1
          .long 0xffffffff        # b[1] = -1
          .long 0xffffffff        # b[2] = -1
          .long 0xffffffff        # b[3] = -1
          .long 0xffffffff        # b[4] = -1
          .long 0xffffffff        # b[5] = -1
          .long 0xffffffff        # b[6] = -1
          .long 0xffffffff        # b[7] = -1
          .long 0xffffffff        # b[8] = -1
          .long 0xffffffff        # b[9] = -1
```

Recall our architecture:



- 1 byte is stored at each address in memory
- Registers hold data the size of an int and the size of an address (32 bits/4 bytes)
- Fast access (roughly single cycle access)
- instructions, except load and store, operate on data that is in registers


Simple Machine (SM213-Solution) - example_global_static_starter.s
Open... Save Save As... Reset Data Checkpoint Data Run Step Run Slowly Halt Show Animation Pause Slower Faster
Register File
Reg Value
r0: 00000000
r1: 00000000
r2: 00000000
r3: 00000000
r4: 00000000
r5: 00000000
r6: 00000000
r7: 00000000
Reg Views
As Int As Ref
0
0
0
0
0
0
0
0
Memory - 100
Addr 0 1 2 3
0x100: 00 00 00 00
0x104: 00 02 01 00
0x108: 00 00 10 00
0x10c: 30 01 10 12
0x110: 03 00 00 00
0x114: 20 00 42 32
0x118: f0 00 00 00
Instructions - 100
B Addr Mac Label Asm Comment
0x100: 00-- 00000002 ld $0x2, r0 r0 = 2
0x106: 01-- 00001000 ld $a, r1 r1 = address of a
0x10c: 3001 st r0, 0x0(r1) a = value in r0 (2)
0x10e: 1012 ld 0x0(r1), r2 r2 = value at address of a
0x110: 03-- 00002000 ld $b, r3 r3 = address of b
0x116: 4232 st r2, (r3, r2, 4) b[a] = a
0x118: f0-- halt halt
Memory - 1000
Addr 0 1 2 3
0x1000: 00 00 00 04
Data - 1000
As Int As Ref Label Comment
4 a a = 4
Memory - 2000
Addr 0 1 2 3
0x2000: ff ff ff ff
0x2004: ff ff ff ff
0x2008: ff ff ff ff
0x200c: ff ff ff ff
0x2010: ff ff ff ff
0x2014: ff ff ff ff
0x2018: ff ff ff ff
0x201c: ff ff ff ff
0x2020: ff ff ff ff
0x2024: ff ff ff ff
Data - 2000
As Int As Ref Label Comment
-1 b b[0] = -1
-1 b[1] = -1
-1 b[2] = -1
-1 b[3] = -1
-1 b[4] = -1
-1 b[5] = -1
-1 b[6] = -1
-1 b[7] = -1
-1 b[8] = -1
-1 b[9] = -1
Current Instruction
nop
nop
Reg Value
PC: 00000100
Instruction: 0000 00000000
Ins Op Code: 0
Ins Op 0: 0
Ins Op 1: 0
Ins Op 2: 0
Ins Op Imm: 00
Ins Op Ext: 00000000
File loaded into memory.

C

```
int a;    // an int
int b;    // an int
int c[4]; // array of 4 ints
void foo () {
  c[a] = a + b;
  a = a + 1;
}
```

Machine Code

Simple Machine .s file

ASM Language Specification

Name	Semantics	Assembly	Machine
load immediate	r[d] ← v	ld $v, rd	0d-- vvvvvvvv
load base+offset	r[d] ← m[(o=p*4)+r[s]]	ld 0(rs), rd	1psd
load indexed	r[d] ← m[r[s]+4*r[i]]	ld (rs,ri,4), rd	2sid
store base+offset	m[(o=p*4)+r[d]] ← r[s]	st rs, 0(rd)	3spd
store indexed	m[r[d]+4*r[i]] ← r[s]	st rs, (rd,ri,4)	4sdi
register move	r[d] ← r[s]	mov rs, rd	60sd
add	r[d] ← r[d] + r[s]	add rs, rd	61sd
and	r[d] ← r[d] & r[s]	and rs, rd	62sd
inc	r[d] ← r[d] + 1	inc rd	63-d
inc address	r[d] ← r[d] + 4	inca rd	64-d
dec	r[d] ← r[d] - 1	dec rd	65-d
dec address	r[d] ← r[d] - 4	deca rd	66-d
not	r[d] ← ~ r[d]	not rd	67-d
shift left	r[d] ← r[d] << s	shl $s, rd	71ss
shift right	r[d] ← r[d] >> -s	shr $s, rd	
halt	halt machine	halt	F0--
nop	do nothing	nop	FF--