ECED 4260: IC Design and Fabrication

The S-Machine Instruction Set Architecture

Larry Hughes, PhD
Electrical and Computer Engineering
Dalhousie University

19 October 2023
Revised: 20 Oct 23 and 23 Oct 23

Introduction

This document describes the Instruction Set Architecture (ISA) of a simple Central Processing Unit (CPU), the S-Machine.

The S-Machine has a limited instruction of 15 instructions (see Table 3). It is a load-store machine, requiring registers to be loaded with data before an operation can take place.

Each instruction is 16-bits wide with a 4-bit opcode. The machine's memory is limited to 256 words, where a word is 16-bits. Data memory size can be doubled to a staggering 512 words by modifying the load and store instructions. Code memory size is limited to 256 words.

The machine is a RISC architecture and allows either a Harvard or Princeton memory organization.

Addressing is absolute and cannot be easily modified during a program's execution. Branching, both unconditional and conditional uses 8-bit absolute addressing.

CPU registers and status bits

The machine has two data registers, a program counter, and three status bits.

The data registers

The S-Machine has two data registers, register A (RA) and register B (RB).

Data register A (RA) is used in all arithmetic operations. The result of an operation is stored in register A.

Register B is a temporary register

All operations are defined as:

`RA ← RA <operand> RB`

Where `<operand>` is one of six operations (`add`, `subtract`, `or`, `and`, and `exclusive-or`).

The program counter

The program counter (PC) is an 8-bit register. It contains the address of the *next* instruction to be executed. The default boot address is `0x00`.

The PC is not directly accessible by the programmer (like RA and RB are). It is incremented by 1 after each instruction is fetched.

The flow-of-control can be changed using the branch instruction.

If the PC reaches the end of program memory (address `0x00FF`), it wraps to address `0x0000`).

The status bits

The S-Machine has three status bits:

Z: The value in RA is zero after an operation has occurred.

N: The value in RA is negative (i.e., RA bit <15> is set) after an operation has occurred.

C: A carry has occurred after an operation has occurred. The truth table to detect carry is shown in Table 1.

Table 1: Carry bit combinations after arithmetic operation

RA <15> (Before)	RB <15>	RA <15> (Result)	Carry
0	0	0	0
0	0	1	0
0	1	1	0
0	1	0	1
1	0	1	0
1	0	0	1
1	1	0	1
1	1	1	1

Any or all these bits can be set or cleared using the SET and CLR instructions.

The compare instruction, CMP, also sets or clears bits N and Z.

Instruction cycle

The S-machine has a three stage or phase instruction cycle: Fetch, Decode, and Execute. These phases are executed in this order continuously.

Fetch

The fetch phase fetches the next instruction to be executed, specified by the PC. After the instruction is fetched, the PC is incremented by 1.

Decode

In this phase, the CPU decodes the instruction into is opcode (bits <15:12>) and, depending on the instruction, into its operands. This information is used to setup the execute phase.

Execute

The action associated with the execution cycle depends on the instruction, as shown in Table 2.

Table 2: Execution phases

Instruction bits <15:14>	Description
00	Instructions that have operands and 8-bit addresses or data values.
01	Arithmetic and logic instructions without operands.
10	
11	CPU status bit setting and clearing instructions. (CMP is a member of this group, it should be with arithmetic and logic instructions.

Instructions

The S-machine has 15 instructions, they are described here.

LD: Load

Load either register A or B with either a 16-bit value from memory or an 8-bit immediate value from the instruction. The 8-bit immediate value can be copied into the target register's high or low byte.

The format of the instruction is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	R	L	B	0	D	D	D	D	D	D	D	D

R: Register, either RA (0) or RB (1).

L: Location of data, either a direct value in memory (0) or an immediate value in the instruction (1).

B: Byte, if the data is an immediate value (L = 1), the byte (bits <7:0>) is stored in either the low byte (0) or the high byte (1).

D: Bits <7:0>, either the address of data to be read or immediate value to load.

ST: Store

The store instruction stores the 16-bit value from register A or B in the memory location specified in bits <7:0>.

The format is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	R	0	0	0	A	A	A	A	A	A	A	A

Where R indicates a register, either A (0) or B (1) and A (bits <7:0>) specify the location in memory to where the register value is to be written.

INC: Increment

The increment instruction increments one of the registers (A or B) by a one-byte value.

The format of the instruction is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	R	0	0	0	V	V	V	V	V	V	V	V

Where R is the register, either A (0) or B (1). The value, V, to be added to the register is stored in bits <7:0>.

The Z, N, and C bits should be set or cleared after INC has executed.

BR: Branch (Conditional and unconditional)

Branch or change the flow of control to the specified address, either conditionally or unconditionally. The instruction is microcoded to support: branch always (BR), branch if zero (BZ), branch if negative (BN), or branch if carry (BC), branch if not zero (BNZ), branch if not negative (BNN), or branch if not carry (BNC).

The format of the instruction is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	X	Z	N	C	A	A	A	A	A	A	A	A

Where X denotes whether the branch is uncomplemented (i.e., BZ, BN, or BC) or uncomplemented (BNZ, BNN, or BNC). If all four bits are set or clear, the branch always occurs (BR). The Z bit is checked if bit<10> is set, the N bit is checked if bit<9> is set, and the C bit is checked if bit<8> is set.

The pseudocode for the instruction is:

```
Execute ← 0
IF Instruction . bit<11> = 0 THEN
      * BZ, BN, BC plus others
      IF Instruction . bit<10> = Z.bit OR
         Instruction . bit<9> = N.bit OR
         Instruction . bit<8> = C.bit THEN Execute ← 1
      ELSE
      * 0 0 0 0
      IF Instruction . bit<10> = 0 AND
         Instruction . bit<9> = 0 AND
         Instruction . bit<8> = 0 THEN Execute ← 1
      ENDIF
ELSE
      * BNZ, BNN, BNC
      IF (Instruction . bit<10> = 1 AND Z.bit = 0) OR
         (Instruction . bit<9> = 1 AND N.bit = 0) OR
         (Instruction . bit<8> = 1 AND C.bit = 0) THEN Execute ← 1
      ENDIF
      * 1 1 1 1
      IF Instruction . bit<10> = 1 AND
         Instruction . bit<9> = 1 AND
         Instruction . bit<8> = 1 THEN Execute ← 1
      ENDIF
ENDIF
IF Execute = 1 THEN
      PC ← Instruction . bit<7:0>
ENDIF
```

The status bits remain unchanged after BR is executed.

ADD: Add

The Add instruction adds RB to RA with the result being stored in RA.

There are no operands associated with the instruction. The format of the instruction is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The Z, N, and C bits should be set or cleared after ADDA has executed.

SUB: Subtract

The Sub instruction subtracts RB from RA with the result being stored in RA.

There are no operands associated with the instruction. The format of the instruction is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

The Z, N, and C bits should be set or cleared after SUBA has executed.

OR

The OR instruction or's the contents of RB with RA with the result being stored in RA.

There are no operands associated with the instruction. The format of the instruction is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

The Z and N bits should be set or cleared after OR has executed. The C bit should be cleared.

AND

The AND instruction and's the contents of RB with RA with the result being stored in RA.

There are no operands associated with the instruction. The format of the instruction is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

The Z and N bits should be set or cleared after AND has executed. The C bit should be cleared.

XOR: Exclusive OR

The XOR instruction exclusive or's the contents of RB with RA with the result being stored in RA.

There are no operands associated with the instruction. The format of the instruction is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The Z and N bits should be set or cleared after XOR has executed. The C bit should be cleared.

SHR: Shift right

This instruction shifts the bits in RA one-bit to the right. The C bit receives the value of bit<0>. Bit<15> is assigned zero. The pseudocode is as follows:

```
C bit ← Bit<0>
Bit<0> ← Bit <1>
…
Bit<14> ← Bit<15>
Bit<15> ← 0
```

The format of the instruction is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

TO shift right, add RA to RA.

MOV: Move RA to RB

The contents of RA is moved to RB. The format of the instruction is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

The status bits are not changed.

EXCH: Exchange RA and RB

The contents of the RA and RB registers are exchanged (swapped). The pseudocode is as follows (the Temp register is internal and cannot be accessed by the programmer.

```
Temp ← RA
RA ← RB
RB ← Temp
```

The instruction format is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

The status bits are not changed.

CMP: Compare

The compare instruction compares RA and RB by subtracting RB from RA. The result of the operation is discarded.

The instruction format is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Status bits Z, N, and C can be set or cleared in this operation.

SET: Set one or more status bits

One or more of the status bits (bits Z, N, and C) are set. The pseudocode is:

```
IF Instruction . bit<10> = 1 THEN Z ← 1
IF Instruction . bit<9> = 1 THEN N ← 1
IF Instruction . bit<8> = 1 THEN C ← 1
```

The instruction format is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	Z	N	C	0	0	0	0	0	0	0	0

CLR: Clear one or more status bits

One or more of the status bits (bits Z, N, and C) are set. The pseudocode is:

```
IF Instruction . bit<10> = 1 THEN Z ← 0
IF Instruction . bit<9> = 1 THEN N ← 0
IF Instruction . bit<8> = 1 THEN C ← 0
```

The instruction format is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	Z	N	C	0	0	0	0	0	0	0	0

Table 3: S-Machine ISA

Opcode	Operands	Operation	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LD	A|B; D|I; H|L; Address | Value	Load A|B Direct | Immediate value Immediate: High | Low byte	0	0	0	0	R	A	B	0	D	D	D	D	D	D	D	D
ST	A|B; Address	Store A | B	0	0	0	1	R	0	0	0	A	A	A	A	A	A	A	A
INC	A|B; Value	Increment A |B by immediate value	0	0	1	0	R	0	0	0	V	V	V	V	V	V	V	V
BR	X, Z, N, C; Address	PC <- Address conditional on XZNC X=0: Check Z, N, C X=1: Check NZ, NN, NC X=Z=N=C=0|1: Unconditional branch	0	0	1	1	X	Z	N	C	A	A	A	A	A	A	A	A
ADD	No operand	RA <- RA + RB	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SUB	No operand	RA <- RA - RB	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
OR	No operand	RA <- RA OR RB	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
AND	No operand	RA <- RA AND RB	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
XOR	No operand	RA <- RA XOR RB	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SHR	No operand	RA <- RA >> 1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
MOV	No operand	RB <- RA	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
EXCH	No operand	RB <-> RA	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
CMP	No operand	Z, N, C <- RA - RB	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SET	Z, N, C	Set one or more of Z, N, C Z=1: Z bit <- 1 N=1: N bit <- 1 C=1: C bit <- 1	1	1	0	1	0	Z	N	C	0	0	0	0	0	0	0	0
CLR	Z, N, C	Clear one or more of Z, N, C Z=0: Z bit <- 0 N=1: N bit <- 0 C=1: C bit <- 0	1	1	1	0	0	Z	N	C	0	0	0	0	0	0	0	0

Errata

20 Oct 23: CMP instruction included status bits. It should not have the status bits.

22 Oct 23: Addition bit combinations added to Table 1.

22 Oct 23: Missing ENDIF in branching pseudocode.