The W10-560 Machine's Characteristics

Memory

The machine has 256 20-bit words, with addresses 0-255.

Registers

Program Counter (PC register) - 20 bits (19 data bits, high order sign bit always positive); contains the address of the next instruction to be fetched for execution (assuming the current instruction is not a branch). Put another way, for *all* instructions, the value of the PC register is 1 greater than the address of the instruction currently executing.

4 Registers (named R0-R3) - 20 bits each; used for computation and indexing (except that R0 cannot be used for indexing).

Arithmetic

Arithmetic provided by this machine is fixed-point only (i.e., no floating-point arithmetic). All integers (zero, positive, and negative) are represented in two's complement form.

Defining integer division carefully is particularly important, as some definitions are different than others; i.e., some machines give different results than others. For this machine, we define integer division (/) in terms of multiplication and addition. We begin with the case of a nonnegative dividend and a positive divisor. Let n be the numerator (or dividend) and d be the denominator (or divisor), with $0 \leq n$ and $0 < d$. Let n, d, q, the quotient, and r, the remainder, all be integers. Then we define / by

$$n/d = q, \text{ where } n = q \times d + r \text{ and } 0 \leq r < d.$$

The cases of negative dividends and/or divisors are defined in terms of the nonnegative case. If $0 \leq n$ and $d < 0$ or $n \leq 0$ and $0 < d$, then

$$n/d = -(|n|/|d|).$$

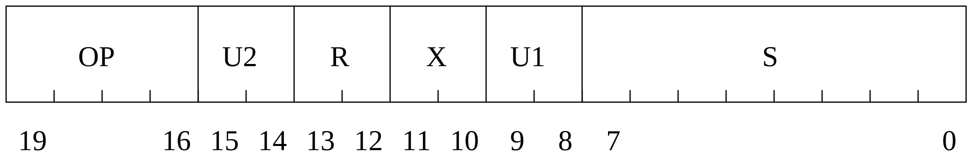
If $n \leq 0$ and $d < 0$, then

$$n/d = |n|/|d|.$$

Consequently, we have, for example, $9/3 = 3$, $(-9)/3 = -3$, $9/(-3) = -3$, $(-9)/(-3) = 3$, $8/3 = 2$, $(-8)/3 = -2$, $8/(-3) = -2$, and $(-8)/(-3) = 2$.

Leaving behind the discussion of division, we discuss overflows generally. Overflows (positive and negative) can occur, for example 0x7FFFF + 0x7FFFF (i.e., 524,287 + 524,287) yields 0xFFFFE (i.e., -2). In a second example, 0x80000 / 0xFFFFF (i.e., (-524,288) / (-1)) yields 0x80000 (i.e., -524,288). There is no other example of division overflowing. The preceding two examples show what the machine always does in those two circumstances. In other circumstances, the machine behaves consistently; i.e., it behaves similarly in similar circumstances. An example overflow from multiplication is: 0x7FFFE * 0x18 (i.e., 524,286 * 24) yields 0xFFFD0 (i.e., -48). A second example overflow from multiplication is: 0x60000 * 0x18 (i.e., 393,216 * 24) yields 0x0 (i.e., 0). Subtraction, of course, can overflow, too. The machine gives no special indication of overflow; it simply supplies predictable results consistent with the examples above.

Instruction Format



OP bits 19 through 16

U2 bits 15 through 14

R bits 13 through 12

X bits 11 through 10

U1 bits 9 through 8

S bits 7 through 0

OP is the "operation field" giving the opcode. U2 is an unused field; the instruction is invalid, and is, therefore, a no-op, if either bit of this field is not zero. R indicates the register involved in the instruction, or different modes for the branch and IO instructions. The X field indicates if indexing is involved in the instruction (a zero value indicates no indexing). U1 is an unused field; if either bit of this field is not zero, the instruction is a no-op. Finally, S is the value to be used in computing the effective address or shift amount.

In the machine instruction descriptions that follow, the notation S(X) will denote the following value:

> If X = 0, then S(X) = nonnegative integer value of the S field (encoded there in simple binary notation; note that $0 \leq S \leq 255$). Otherwise, S(X) = result of adding the value of the S field to register X. The value of register X does not change as a result of this computation. Note that this can result in a negative number or a value > 255. If the value is not used as an immediate value but as an address and that address is outside of the range $0 \leq S(X) \leq 255$, an "Address Range" error occurs. This error is not fatal, and the instruction causing the error is treated as a no-op.

Machine Instructions

Hex Opcode	Assembler Instruction		Description
0	LD	R,S(X)	Load. R ← contents of address S(X).
1	LDI	R,S(X)	Load Immediate. R ← S(X)
2	ST	R,S(X)	Store. Contents of R is stored in address S(X).
3	ADD	R,S(X)	Add. R ← R + contents of address S(X).
4	SUB	R,S(X)	Subtract. R ← R - contents of address S(X).
5	MUL	R,S(X)	Multiply. R ← R * contents of address S(X).
6	DIV	R,S(X)	Divide. R ← R / contents of address S(X). (Note that integer division is done here; only the quotient is stored in register R. An attempt to divide by zero is an error condition.)
7	OR	R,S(X)	Or. R ← the bit-wise **or** of R with the contents of address S(X).
8	AND	R,S(X)	And. R ← the bit-wise **and** of R with the contents of address S(X).
9	SHL	R,S(X)	Shift Left. Shift register R to the left by S(X) places. S(X) must have a value in the range [0, 19]. This shift is “arithmetic”. Thus, the bits introduced at the low end are all zero.
A	SHR	R,S(X)	Shift Right. Shift register R to the right by S(X) places. S(X) must have a value in the range [0,19]. This shift is “arithmetic”. Thus, the value of the sign bit originally in R is introduced at the high end.

Machine Instructions continued:

Hex Opcode	Assembler Instruction	Description
B	IO R,S(X)	Input/Output. A *byte* here means a nonnegative integer value b where $0 \leq b < 2^8$; hence, b is a byte if and only if b can be represented by 8 bits using the simple binary encoding scheme. Each of input and output is considered a stream of bytes. Let b be known as a *separator byte* if and only if either $0 \leq b \leq 32$ or $127 \leq b \leq 160$. R $= 0 \Rightarrow$ Consume from the input stream all separator bytes at the front of the stream. Then consume from the input stream the “word” consisting of the longest sequence of non-separator bytes up to the end of the stream or the next separator byte; also consume this separator byte. If the sequence of characters obtained from the “word” according to the ASCII encoding scheme is not a well-formed decimal representation (optionally with ‘-’ or ‘+’ as the first byte) of a number representable in a 20-bit 2’s complement encoding, then make no changes to register or memory values; otherwise, place the represented value in address S(X). R $= 1 \Rightarrow$ Consume just the first byte, b, at the front of the input stream. Replace the least significant 8 bits of register X with b. Leave the remaining bits of X alone. The S field is ignored. R $= 2 \Rightarrow$ Extend the output stream with the sequence of bytes that, according to the ASCII encoding scheme, is the well-formed character-string decimal representation of the value encoded at address S(X) according to the 20-bit 2’s complement scheme. Do not send to output any separator bytes or the ‘+’ byte. R $= 3 \Rightarrow$ Extend the output stream with the single byte that is encoded in the 8 most significant bits of register X according to the simple binary scheme. The S field is ignored.

Machine Instructions continued:

Hex Opcode	Assembler Instruction		Description

C	BR	R,S(X)	Branch unconditional.
			R=0 $\Rightarrow$ halt (S ignored)
			with X=0, quiet;
			with X=1, dump (i.e., *display*, but neither reset nor erase) all of memory;
			with X=2, dump all registers (and PC);
			with X=3, dump both memory and registers (and PC);
			R=1 $\Rightarrow$ dump all registers, PC, and memory and branch to address S(X)
			R=2 $\Rightarrow$ branch to address PC + S(X)
			R=3 $\Rightarrow$ branch to address S(X)

D	BRZ	R,S(X)	Branch if Zero. If contents of register R = 0, then branch to address S(X).
E	BRN	R,S(X)	Branch if Negative. If contents of register $R < 0$, then branch to address S(X).
F	BRS	R,S(X)	Branch to Subroutine. R $\leftarrow$ PC; branch to address S(X).

Note: Error conditions, such as shift by more than 19 bits, and divide by zero, are NOT fatal. They result in no-ops, and execution proceeds with the instruction at the next sequential memory location. I/O errors (e.g., reading past end-of-file) can be fatal, if you so desire.

[This page was left blank intentionally.]