Stalling 2012 edition

CS 320
Ch. 12 Instruction Sets

Computer instructions are written in mnemonics. **Mnemonics typically have a 1 to 1 correspondence between a mnemonic and the machine code. Mnemonics are the assembly language instructions.**

A computer instruction typically consists of an operation code (opcode), source operand, and destination operand. **Opcode tells what the instruction does and is essential. Source and destination operands tell what the data is that the instruction is to operate on. Operands are optional.**

When an instruction is initially read from memory it goes to the **Instruction register.**

There are four categories of assembly language instructions: **Data Processing (ALU), Data Storage(Memory), Data Movement(I/O), Control(branch and others)**

We can typically associate five addresses with a single assembly language instruction: **address of instr, addr os next instr, addr of src operand, addr of dest operand, addr of 2nd src operand.**

An addressing mode is the m**ethod whereby the operand specifies the data.**

Machine instructions, assembly language instructions, and instructions in a HLL are related like this:
HLL → compiler →assembly instructions →machine code.
HLL:Assembly is many to one, Assembly: machine code is 1:1

Some instructions may have four direct addresses associated with them. These include 2 operands, 1 result operand, and address of next instruction. For example Add A, B, C which adds B to C and stores the result in A. These instructions are not common because t**he number of operands makes the instruction format long.**

A typical 2-operand instructions such as Add A, B works like this: **A ← A + B**

A 1-operand instruction works like this: **Add B A ← A + B. Accumulator is an implied operand.**

The advantage of 1-operand instructions is that they are s**hort and fast. The disadvantage is that they may force many operations to require two or more instructions. A is overused.**

In term of instruction set design trade-offs with regard to number of operands: We w**ant a short instruction for speed and space but need a long instruction for convenience and usability.**

The author lists 5 fundamental design issues with respect to instruction set design. They are:
Operation repertoire – how many and which operations to provide

Stalling 2012 edition

Data types – types and length of data to be operated on
Instruction Format – instruction length, number of addresses, field size
Registers – number of CPU registers
Addressing – addressing modes available.

The four most important general categories of data are: **Addresses, numbers, characters, and logical data.**
Addresses are considered to be a data type because of **pointers**

Other data types that might be used include: **bits, strings, objects**

With regard to numbers a packed decimal represents the d**ecimal digits 0 to 9 as 4 bits with two digits per byte or 8 digits in a 32 bit word.**

Note that Ints are the counting numbers 1, 2, 3, etc. Ints are also fixed point. But fixed point just means that the decimal point is fixed at a specific location. For example the currency type has the decimal at two places.

IRA, ASCII, and EBCDIC are **character coding schemes. IRA → International Reference Alphabet, ASCII → American Standard Code for Information Interchange, EBCDIC → Extended Binary Coded Decimal Interchange Code from IBM. IRA and ASCII are the same.**

Logical data is a **bit-oriented view of data. Used most commonly for true and false Boolean values.**

Intel x86 and ARM Operations
The Intel 8086 was created in 1978. It remains a relevant architecture today because Intel always makes the next processor backward compatible. In addition, the x86 architectures is n**ow used on ATOM.**

The *general* data types on the x86 are: **byte, 16-bit word, 32-bit doubleword, 64-bit quadword, and 128 bit double quad word.**

The general data types on the ARM? **byte, 16-bit half word, 32 bit word.**

Big endian and little endian: **Little endian stores the least significant byte at the lowest address. Big endian stores the most significant byte at the lowest address. Intel is little endian, Motorola is big endian, the ARM is ambidexterous. Little endian has some advantage with respect to doing multibyte arithmetic but it's minor.**

The terms little endian and big endian come from **Gulliver's travels in which there is a religious war between two groups who break eggs on either the little end or the big end. (And they say geeks have no sense of humor).**

Stalling 2012 edition

The author lists 7 generic instruction types. Here they are with examples:

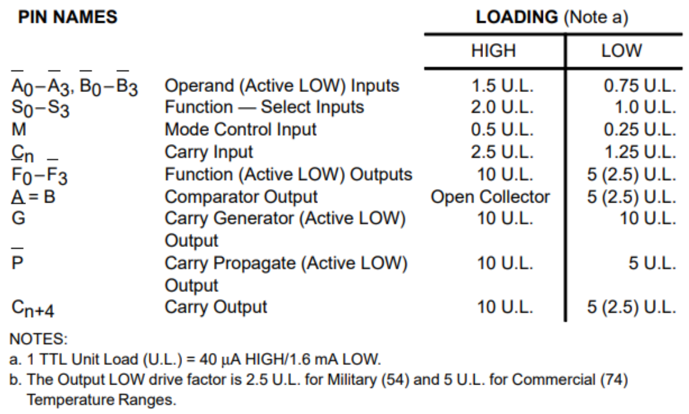
- **Data transfer → move a, b**
- **Arithmetic → add a, b**
- **Logical → and a, b**
- **Conversion → translate**
- **I/O → out port, a**
- **System Control → hlt**
- **Transfer of control → jmp target**

Most processors have a shift instruction. A few have a rotate and some have both a rotate and a shift. The difference between a rotate and a shift is that i**n rotate what falls out of one end gets put back in the other. In shift what falls out is lost. The 8051 has only rotates and no shifts. Either can be done without the other.**

Some processors also have a skip and a jump: **skip skips just one instruction based on a condition. jump jumps anywhere.**

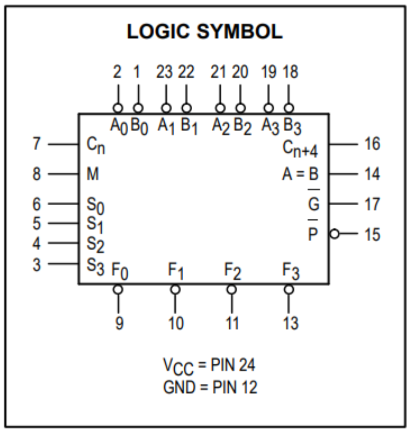
Branches are often conditional.   **Condition codes are typically set by the ALU and saved as flags in a register. They indicate the result of a previous operation.**

This is the 74181 4-bit ALU. The lines F_0 to F_3 are the flags

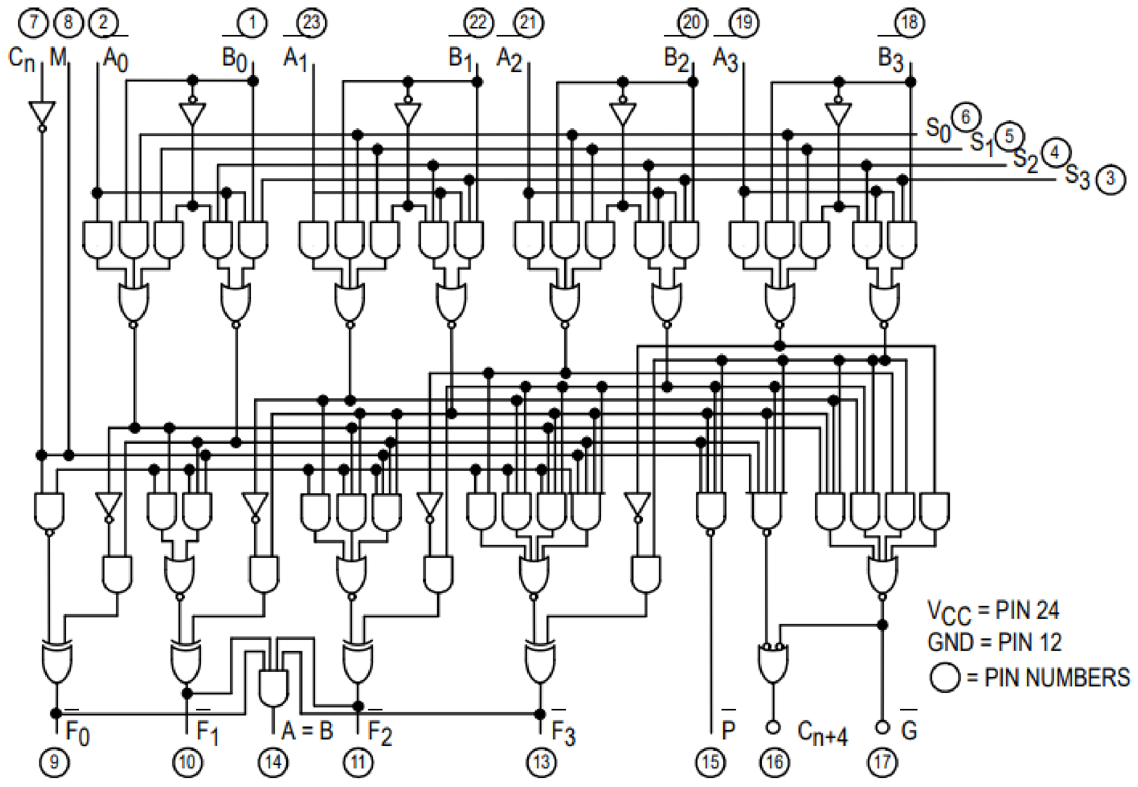


PIN NAMES		LOADING (Note a)	
		HIGH	LOW
$\overline{A}_0$–$\overline{A}_3$, $\overline{B}_0$–$\overline{B}_3$	Operand (Active LOW) Inputs	1.5 U.L.	0.75 U.L.
S_0–S_3	Function — Select Inputs	2.0 U.L.	1.0 U.L.
M	Mode Control Input	0.5 U.L.	0.25 U.L.
C_n	Carry Input	2.5 U.L.	1.25 U.L.
$\overline{F}_0$–$\overline{F}_3$	Function (Active LOW) Outputs	10 U.L.	5 (2.5) U.L.
A = B	Comparator Output	Open Collector	5 (2.5) U.L.
$\overline{G}$	Carry Generator (Active LOW) Output	10 U.L.	10 U.L.
$\overline{P}$	Carry Propagate (Active LOW) Output	10 U.L.	5 U.L.
C_{n+4}	Carry Output	10 U.L.	5 (2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.



Stalling 2012 edition



FUNCTION TABLE

MODE SELECT INPUTS				ACTIVE LOW INPUTS & OUTPUTS		ACTIVE HIGH INPUTS & OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M = H)	ARITHMETIC** (M = L) (C_n = L)	LOGIC (M = H)	ARITHMETIC** (M = L) (C_n = H)
L	L	L	L	A	A minus 1	A	A
L	L	L	H	AB	AB minus 1	A + B	A + B
L	L	H	L	A + B	AB minus 1	AB	A + B
L	L	H	H	Logical 1	minus 1	Logical 0	minus 1
L	H	L	L	A + B	A plus (A + B)	AB	A plus AB
L	H	L	H	B	AB plus (A + B)	B	(A + B) plus AB
L	H	H	L	A ⊕ B	A minus B minus 1	A ⊕ B	A minus B minus 1
L	H	H	H	A + B	A + B	AB	AB minus 1
H	L	L	L	AB	A plus (A + B)	A + B	A plus AB
H	L	L	H	A ⊕ B	A plus B	A ⊕ B	A plus B
H	L	H	L	B	AB plus (A + B)	B	(A + B) plus AB
H	L	H	H	A + B	A + B	AB	AB minus 1
H	H	L	L	Logical 0	A plus A*	Logical 1	A plus A*
H	H	L	H	AB	AB plus A	A + B	(A + B) plus A
H	H	H	L	AB	AB plus A	A + B	(A + B) Plus A
H	H	H	H	A	A	A	A minus 1

L = LOW Voltage Level
H = HIGH Voltage Level
*Each bit is shifted to the next more significant position
**Arithmetic operations expressed in 2s complement notation

Stalling 2012 edition

Most processor have both a call and a jump. C**all saves PC on stack in addition to doing a jump.**

Return addresses for calls may be stored on one of three locations. They are: **register, start of procedure, or top of stack.**

A reentrant procedure is o**ne which can be recalled while running by an interrupt without loss of data.**

A stack is a **LIFO buffer.**

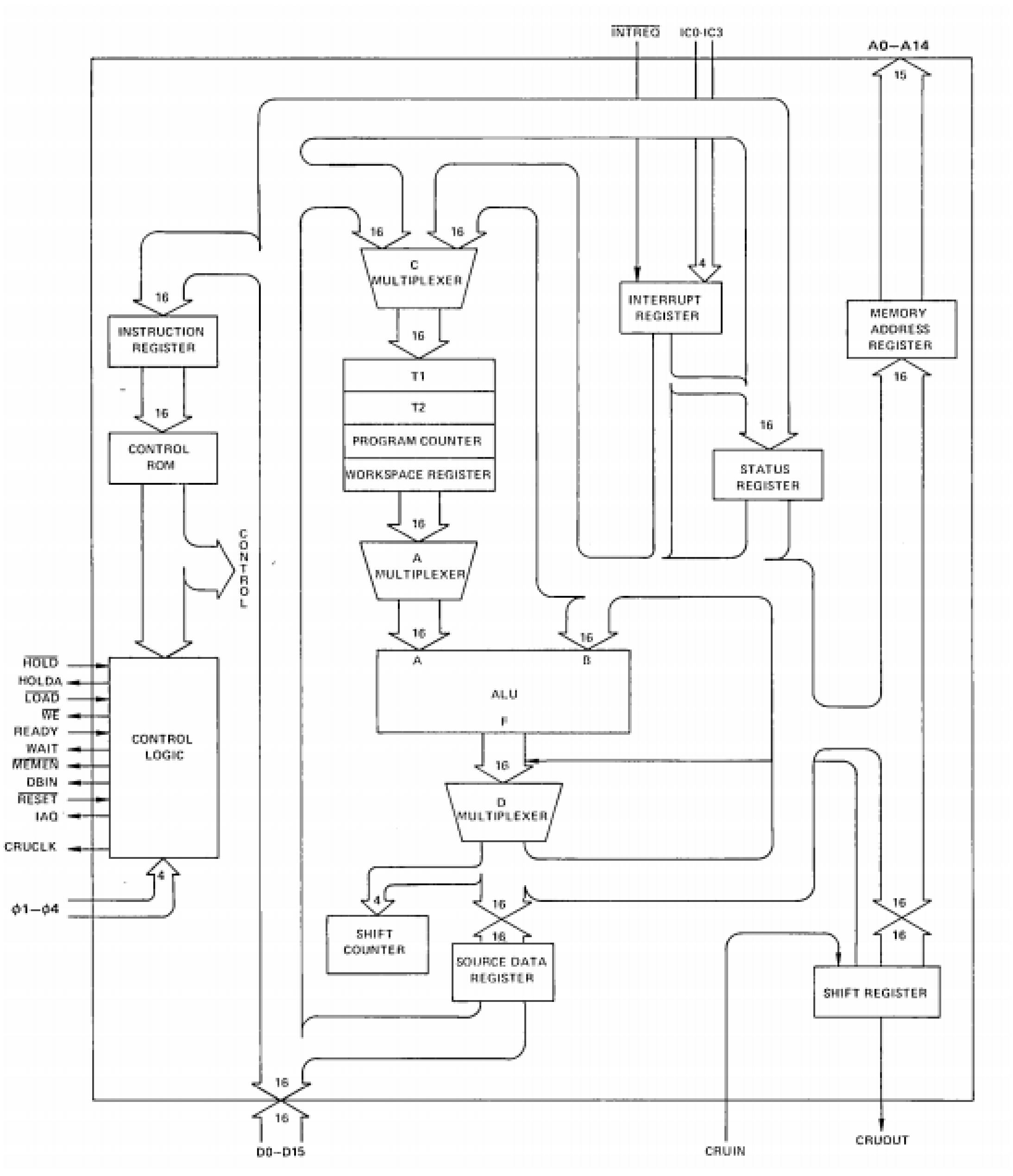
Push and Pop are zero address instructions. **There is no address with the instruction. Instead the address is the one pointed to by the stack pointer register.**

Three addresses are used for stack operation. These are often stored in special CPU registers. The stack registers are: **Stack pointer, stack base, stack limit. SP holds the address of the top of the stack, SB is the address of the start of the stack and is used to determine when the stack is empty, SL is address of the bottom of the stack and is used to prevent stack overflow.**

Only the stack save method is reentrant. Almost all processors have a stack.

The only processor I know of that did not have a stack was the Texas Instruments TMS 9900.

Stalling 2012 edition



TMS 9900

The workspace register pointed to a location in memory which had 16, 16-bit locations called "registers". To write a reentrant program a stack had to be created in software.

In addition to saving return addresses m**any HLLs use the stack to pass parameters to procedures.**

A stack frame is t**he set of addresses and data that is stored on the stack by or for a procedure.**

Stalling 2012 edition

The x86 architecture has two different sets of logical compare instructions. One set tests for less than or greater than and the other tests for below or above. The difference between these two is that b**elow and above are for unsigned arithmetic.**

The x86 has four instructions to support procedure calls. These are: **Call, Return, Enter (Creates a stack frame for block structured HLLs such as COBOL, PASCAL, and ADA), Leave (reverses Enter).**

The x86 and ARM have a carry flag and an auxiliary Carry flag. The auxiliary carry flag? **Carry from bit 3 to bit 4 (half carry). It's used for BCD arithmetic.**