Ternary Emulation Manual

Scott Carda

Contents

Introduction

Ternary Logic

The Emu Emulator

Heptavigesimal Numbering System

Instruction Format

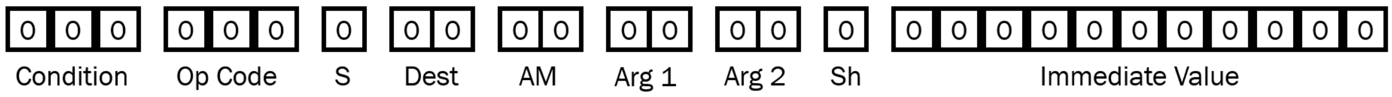

Figure 1: 27-trit Instruciton Format

Condition: The 3-trit condition code the machine uses to determine whether the instruction is executed.

Op Code: The 3-trit code that indicates which operation is performed with this instruction.

S: The instruction's flag mode. The jump operation has unique behavior associated with this.

Dest: The 2-trit indicator for the destination register.

AM: The instruction's 2-trit address mode.

Arg 1: The 2-trit indicator for the register to be used as the first argument to the operation.

Arg 2: The 2-trit indicator for the register to be used as the second argument to the operation.

Sh: The instruction's shift mode. Indicates which shift operation is performed on Arg 2.

Immediate Value: This 11-trit field is reserved for embedding numerical values in the instruction.

Conditions

The Current Program Status Register or CPSR is a special 3-trit register that keeps track of several flags. There are three flag, one associated with each of the three trits in the register. The most significant trit is the **S** flag, which indicates the sign of the result of an operation. Table 1 shows the meaning of the values for the **S** flag. The middle of the three trits is the **C** flag, which contains the value of any unsigned overflow resulting from an operation. The least significant trit is the **V** flag, which contains the value of any signed overflow resulting from an operation.

Instruction are conditionally executed based on the values of these flag. Common conditions have been abstracted from these flag and made available to the system. Each condition is assigned a 3-trit code for use in the most significant field of an instruction. If the condition is false, the rest of the instruction is ignored, and the machine will move on to the next instruction in its RAM without executing the encoded operation. The conditions available to the system are found in Table 2.

Table 1:

S	Meaning
0	Result was Positive
1	Result was Zero
2	Result was Negative

Table 2:

Condition Code	Meaning	Flags
000	Always	Always
001	Equal	**S** = 1
002	Carry Clear	**C** = 0
010	Negative	**S** = 0
011	Overflow Clear	**V** = 0
012	Unsigned Lower/Equal	**C** = 0 **OR S** = 1
020	Less Than	IS_F(**S**) = IS_F(**V**)
021	Less Than or Equal To	**S** = 1 **OR** IS_F(**S**) = IS_F(**V**)
022-200	*bad condition code*	*bad condition code*
201	Greater Than	**S** = 1 **AND** IS_F(**S**) $\neq$ IS_F(**V**)
202	Greater Than or Equal To	IS_F(**S**) $\neq$ IS_F(**V**)
210	Unsigned Higher	**C** $\neq$ 0 **AND S** $\neq$ 1
211	Overflow Set	**V** $\neq$ 0
212	Positive	**S** = 2
220	Carry Set	**C** $\neq$ 0
221	Not Equal	**S** $\neq$ 1
222	Never	Never

Operations

Flag Mode

Address Mode

Shift Mode

Immediate Value

Raw

27 trits xxx xxx x xx xx xx xx x xx-xxxxxxxxx cond opcode s arg1 AM arg2 arg3 shf val dest opr1 opr2

Conditions:

000: Always 001: Equal (S flag = 1) 002: Carry Clear (C flag = 0) 010: Negative (S flag = 0) 011: Overflow Clear (V flag = 0) 012: Unsigned Lower (C flag = 0, OR S flag = 1) 020: < (IS_F(S flag) = IS_F(V flag)) 021: <= (S flag = 1 OR IS_F(S flag) = IS_F(V flag))

022-200: <bad>

201: > (S flag = 1 AND IS_F(S flag) != IS_F(V flag)) 202: >= (IS_F(S flag) != IS_F(V flag)) 210: Unsigned Higher (C flag != 0 AND S flag != 1) 211: Overflow Set (V flag != 0) 212: Positive (S flag != 0) - maybe: (S flag = 2) 220: Carry Set (C flag != 0) 221: Not Equal (S flag != 1) 222: Never

Shift Values:

0: LSR 1: ASR 2: LSL

Address Modes (AM):

00: Immediate - Data found in Instruction (val) - Ignore opr2 and shf MOV r0, #3 000 022 0 00 00 xx xx x 00-000000010 = 00002200000xxxxx00000000010 ADD r0, r1, #3 000 020 0 00 00 01 xx x 00-000000010 = 0000200000001xxx00000000010

01: Reg Direct - Register of Data found in Instruction (opr2) - Ignore shf and val MOV r0, r1 000 022 0 00 01 xx 01 x xx-xxxxxxxxx = 00002200001xx01xxxxxxxxxxxx

02: Reg Dir Scalled - Register of Data and Scale found in Instruction (val) ADD r0, r1, r2, LSL #15 000 020 0 00 02 01 02 2 00-000000120 = 000020000020102200000000120

10: <bad>

11: Addressed - Address of Data found in Instruction (val) - Ignore opr1, opr2, and shf LDR r0, [#<address>] 000 010 0 00 11 xx xx x ss-<address> = 00001000011xxxxxss<address>

12: Reg Indirect - Address of Data stored in Register found in Instruction (opr1) - Ignore opr2, shf, and val LDR r0, [r1] 000 010 0 00 12 01 xx x xx-xxxxxxxxx = 0000100001201xxxxxxxxxxxxxx

20: -w/ Imm Offset - Reg Indirect Address plus Offset found in Instruction (Reg: opr1, Off: val) - Ignore opr2 and shf LDR r0, [r1, #3] 000 010 0 00 20 01 xx x 00-000000010 = 0000100002001xxx00000000010

21: -w/ Dir Offset - Reg Indirect Address plus Offset found in Reg in Instruction (Reg: opr1, Off: opr2) - Ignore shf and val LDR r0, [r1, r2] 000 010 0 00 21 01 02 x xx-xxxxxxxxx = 000010000210102xxxxxxxxxxx

22: -w/ Scaled Offset - Reg Indirect Address plus Scaled Offset found in Reg in Instruction, Scale is found in Instruction (Reg: opr1, Off: opr2, Scale: val) LDR r0, [r1, r2, LSL #1] 000 010 0 00 22 01 02 2 00-000000001 = 000010000220102200000000001

Opcodes:

(@) 000: HALT (A) 001: NO_OP (B) 002: CLEAR

(C) 010: LOAD (D) 011: STORE (E) 012: EXCHANGE

(F) 020: ADD (G) 021: SUB (H) 022: MOVE

(I) 100: AND (J) 101: OR (K) 102: XOR

(L) 110: F_NOT (M) 111: N_NOT (N) 112: T_NOT

(O) 120: ISF (P) 121: ISN (Q) 122: IST

(R) 200: LSR (S) 201: ASR (T) 202: LSL

(U) 210: SHIFT_M (V) 211: SHIFT_P (W) 212: <bad>

(X) 220: JUMP (Y) 221: <bad> (Z) 222: <bad>