Ternary Emulation Manual

Scott Carda

Contents

Introduction 3

Ternary Logic 3

The Emu Emulator 3

Instruction Format 3

Conditions 3

Operations 4

Flag Mode 4

Address Mode 6

Shift Mode 6

Immediate Value 7

Raw 7

Introduction

Ternary Logic

The Emu Emulator

Heptavigesimal Numbering System

Instruction Format

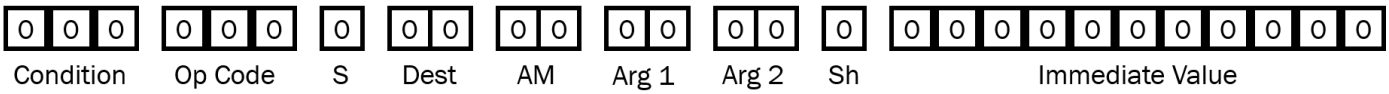

Figure 1: 27-trit Instruciton Format

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

Table 1:

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

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. As this is a 3-trit code, each of these condition codes can be represented as a single heptavigesimal character, which are found in parentheses in the first column of the table.

Table 2:

Condition Code	Meaning	Flags
000 (@)	Always	Always
001 (A)	Equal	$\mathbf{S} = 1$
002 (B)	Carry Clear	$\mathbf{C} = 0$
010 (C)	Negative	$\mathbf{S} = 2$
011 (D)	Overflow Clear	$\mathbf{V} = 0$
012 (E)	Unsigned Lower/Equal	$\mathbf{C} = 0$ **OR** $\mathbf{S} = 1$
020 (F)	Less Than	IS_T($\mathbf{S}$) = IS_F($\mathbf{V}$)
021 (G)	Less Than or Equal To	$\mathbf{S} = 1$ **OR** IS_T($\mathbf{S}$) = IS_F($\mathbf{V}$)
022-200 (H-R)	*bad condition code*	*bad condition code*
201 (S)	Greater Than	$\mathbf{S} \neq 1$ **AND** IS_T($\mathbf{S}$) $\neq$ IS_F($\mathbf{V}$)
202 (T)	Greater Than or Equal To	IS_T($\mathbf{S}$) $\neq$ IS_F($\mathbf{V}$)
210 (U)	Unsigned Higher	$\mathbf{C} \neq 0$ **AND** $\mathbf{S} \neq 1$
211 (V)	Overflow Set	$\mathbf{V} \neq 0$
212 (W)	Positive	$\mathbf{S} \neq 2$
220 (X)	Carry Set	$\mathbf{C} \neq 0$
221 (Y)	Not Equal	$\mathbf{S} \neq 1$
222 (Z)	Never	Never

Operations

Flag Mode

This is a single-trit field in the instruction that denotes the flag mode for the instruction. There are three flag modes that can be used, the value this field takes for each mode is outlined in Table 4. 'No-Set' mode will prevent the instruction from affecting the CPSR register flags. 'Set' mode will allow for the instruction to set the flags in the CPSR register. 'Exclusive-Set' mode will allow for the instruction to set the flags in the CPSR register, while throwing out the usual result of the instruction, instead of storing it to the destination register.

The jump instruction has speacial logic associated with this field. As the jump instruction cannot affect the CPSR flags, the regular flag modes are not applicable. Instead this value is used to determine how to handle capturing the current value in the PC register. If the value of this field is 0, the current

Table 3:

Operation Code	Operation
000 (@)	HALT
001 (A)	NO_OP
002 (B)	CLEAR
010 (C)	LOAD
011 (D)	STORE
012 (E)	EXCHANGE
020 (F)	ADD
021 (G)	SUB
022 (H)	MOVE
100 (I)	AND
101 (J)	OR
102 (K)	XOR
110 (L)	F_NOT
111 (M)	N_NOT
112 (N)	T_NOT
120 (O)	ISF
121 (P)	ISN
122 (Q)	IST
200 (R)	LSR
201 (S)	ASR
202 (T)	LSL
210 (U)	SHIFT_M
211 (V)	SHIFT_P
212 (W)	*bad operation code*
220 (X)	JUMP
221 (Y)	*bad operation code*
222 (Z)	*bad operation code*

Table 4:

S Field	Flag Mode
0	Result was No-Set
1	Result was Set
2	Result was Exclusive-Set

location in code, before the jump, is not recorded, and Destination field is ignored. If the value is 1, the current location in code, before the jump, is stored in the register denoted in the Destination field. This is done to make it easier to implement functions by capturing where the function will return to. If the value of the field is 2, the jump will not affect the PC register, resulting in no jump in code, but will store the current value in the PC register in the register denoted in the Destination field.

Address Mode

There are 8 address modes, which are listed in table Table 5. The first 3 address modes, 00 through 02, are used by non-addressing operations, such as add or sub. The last 5 address modes are used by addressing instructions, such as ldr and str. The address mode dictates the value of the second argument to the operation, or the only argument to the operation for operations that only have one argument. For addressing operations, this value must be an address.

Table 5:

AM Field	Address Mode
00	Immediate
01	Register Direct
02	Register Direct Scaled
10	*bad address mode*
11	Addressed
12	Register Indirect
20	Immediate Offset
21	Direct Offset
22	Scaled Offset

Immediate: The value of the immediate field is used as the argument.

Register Direct: The value of the register indicated by Arg 2 is used as the argument.

Register Direct Scaled: The value of the register indicated by Arg 2 undergoes a shift operation, indicated by the instruction's shift mode, whose magnitude is the value of the immediate field. The result is used as the argument.

Addressed: The value of the immediate field is the address used as the argument.

Register Indirect: The address found in the register indicated by Arg 1 is used as the argument.

Immediate Offset: The sum of the address found in the register indicated by Arg 1 and the value of the immediate field is the address used as the argument.

Direct Offset: The sum of the address found in the register indicated by Arg 1 and the value of the register indicated by Arg 2 is the address used as the argument.

Scaled Offset: The value of the register indicated by Arg 2 undergoes a shift operation, indicated by the instruction's shift mode, whose magnitude is the value of the immediate field. The result is added to the address found in the register indicated by Arg 1, resulting in the address used as the argument.

Shift Mode

This is a single-trit field in the instruction that denotes the flag mode for the instruction. There are three flag modes that can be used, the value this field takes for each mode is outlined in Table 6. There are two address modes which perform a scaling operation, Register Direct Scaled and Scaled

Offset. In these address modes the value of the register indicated by Arg 2 undergoes a shift operation whose magnitude is the value of the immediate field. The shift mode indicates which shift operation is performed during these instructions.

Table 6:

Sh Field	Shift Mode
0	Logical Shift Right
1	Arithmetic Shift Right
2	Logical Shift Left

Immediate Value

The last 11 trits of an instruction represent a numerical value embedded in the instruction. This field is further subdivided into a rotation field, which is the first 2 trits, and the value field, which is the remaining 9 trits. The number that the immediate represents is calculated by treating the value field as the least significant 9 trits in a 27-trit number. That number is then rotated to the right by 3 times the number specified in the rotation field. The resulting number may be interpreted as a signed or unsigned number depending on its use.

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 = 00002200001xx01xxxxxxxxxxx

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 = 00001000022010220000000001

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>