Ternary virtual machine specifications draft FEB 2008

February 25, 2008

Balanced ternary arithmetic, tryte width 6. Address width 12. Standard notation balanced base-9. Accessible address range DDD:DDD$_{b9}$ to 444:444$_{b9}$. Total accessible memory 531$_{10}$ ktrytes. Big endian.

Registers

Name	**Purpose**
A	Accumulator
X	Address register
Y	Address register
PC	Program counter
S	Stack index
CL	Clock register
P	Processor status

Processor status register specification

MST					**LST**
PR	V	B	I	G	C

Where flags have the following purpose

Flag	**Purpose**
C	Carry
G	Comparison. Sign of the result of the last operation
I	No-Interrupt flag
B	Break in progress flag
V	Overflow flag
PR	Parity flag

Note: I and B flags may merge in the future to make better use of ternary capabilities.

Op-code specifications (A address mode, C op-code)

MST					**LST**
A	A	C	C	C	C

Addressing modes

DEC	**B9**	**Symbol**	**Instruction length**	**Description**
-4	D	ABS	3	Value at memory position directly specified
-3	C	IMM	2	Value immediately following op-code
-2	B	AX	3	ABS with offset X
-1	A	AY	3	ABS with offset Y
0	0	ACC	1	Accumulator
0	0	IMP	1	Implicit – no argument
1	1	INDX	3	INDIRECT with offset X
2	2	INDY	3	INDIRECT with offset Y
3	3	INDIRECT	3	Value pointed to by the memory position directly specified
4	4	X:Y	1	Memory at page X, index Y

Operations

DEC	**B9**	**Symbol**	**Valid addressing mods**	**Comment**
-40	DD	CLV	IMP	Clear overflow
-39	DC	BRK	IMP	Trigger interrupt
-38	DB	RTI	IMP	Return from interrupt
-37	DA	LDA	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set accumulator to memory value
-36	D0	LDX	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set Y register to memory value
-35	D1	LDY	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set X register to memory value
-34	D2	STA	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Store accumulator's value in memory
-33	D3	STX	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Store X register's value in memory
-32	D4	STY	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Store Y register's value in memory
-31	CD	TAX	IMP	Transfer A to X
-30	CC	TAY	IMP	Transfer A to Y
-29	CB	TXA	IMP	Transfer X to A
-28	CA	TYA	IMP	Transfer Y to A
-27	C0	TSX	IMP	Transfer Stack index to X
-26	C1	TXS	IMP	Tarnsfer X to Stack index
-25	C2	PHA	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Push to stack
-24	C3	PHP	IMP	Push processor status to stack
-23	C4	PLA	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Pull from stack
-22	BD	PLP	IMP	Pull processor status from stack
-21	BC	AND	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	$A=A \wedge$memory
-20	BB	EOR	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	$A=A \oplus$memory
-19	BA	ORA	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	$A=A \vee$memory
-18	B0	BIT	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set status flag as though AND
-17	B1	ADD	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	$A=A+$memory
-16	B2	CMP	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set status flag as though A-memory
-15	B3	INC	ACC, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Increase value by 1
-14	B4	INX	IMP	Increase X register by 1
-13	AD	INY	IMP	Increase Y register by 1
-12	AC	DEC	ACC,ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Decrease value by 1
-11	AB	DEX	IMP	Decrease X register by 1
-10	AA	DEY	IMP	Decrease Y register by 1
-9	A0	ASL	ACC,ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Shift value left, store spillover in carry
-8	A1	LSR	ACC,ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Shift value right, store spillover in carry
-7	A2	ROL	ACC,ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Rotate left
DEC	**B9**	**Symbol**	**Valid addressing mods**	**Comment**

DEC	**B9**	**Symbol**	**Valid addressing mods**	**Comment**
-6	A3	ROR	ACC,ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Rotate right
-5	A4	JMP	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump
-4	0D	JSR	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump to subroutine (push current PC)
-3	0C	RST	IMP	Return from subroutine (pop PC)
-2	0B	CLC	IMP	Clear carry
-1	0A	CLI	IMP	Clear interrupt flag
0	00	NOP	IMP	No operation
1	01	SEC	IMP	Set carry
2	02	SEI	IMP	Set interrupt flag
3	03	MLH	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A=high tryte(A*memory)
4	04	MLL	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A=low tryte(A*memory)
5	1D	DIV	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A = integer($\frac{A}{memory}$)
6	1C	MOD	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A = remainder($\frac{A}{memory}$)
7	1B	PLX	IMP	Pull X from stack
8	1A	PLY	IMP	Pull Y from stack
9	10	PHX	IMP	Push X to stack
10	11	PHY	IMP	Push Y to stack
11	12	JCC	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if carry is clear
12	13	JCT	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if carry is positive
13	14	JCF	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if carry is negative
14	2D	JEQ	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if comparison flag is clear
15	2C	JNE	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if comparison flag isn't clear
16	2B	JLT	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if comparison flag is negative
17	2A	JGT	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if comparison flag is positive
18	20	JVC	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if overflow is clear
19	21	JVS	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if overflow is set
20	22	IVC	IMP	Invert carry flag
21	23	PRM	IMM, ACC, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Permutate value (tritwise add 1)
22	24	TSH	IMM, ACC, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	PRM without roll over
23	3D	BUT	IMM, ACC, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A BUT Value
24	3C	LAD	ABS, ABSX, ABSY, IND, INDX, INDY	Load ADdress into X:Y
25	3B	RESERVED		
26	3A	RESERVED		
27	30	RESERVED		
28	31	RESERVED		
29	32	RESERVED		
30	33	RESERVED		
31	34	RESERVED		
32	4D	RESERVED		
33	4C	RESERVED		
34	4B	RESERVED		
35	4A	RESERVED		
36	40	RESERVED		
37	41	RESERVED		
38	42	RESERVED		
39	43	RESERVED		
40	44	DEBUG	DEBUG	DEBUG
DEC	**B9**	**Symbol**	**Valid addressing mods**	**Comment**

Reserved addresses

Address range	Purpose
DDD:DDD/C	IRQ, IRQ data
DDD:DDB	Screen mode
DDD:DDA	Disk I/O
DDC:DDD-DDC:444	Stack
DDB:DDD-DDA444	Screen buffer
444:441	Clock interrupt frequency
444:442/3	Interrupt handler vector

Screen

The Screen Mode tryte (DDD:DDB) has the following function

MST					LST
RES	RES	RES	RES	Graphics mode	Redraw

Screen redraws when Redraw is set to 1, and allowed graphics mode are raster(-1), text (0) and vector (1).

Vector mode

Vector mode only uses the first page of the screen buffer (that is, DDB:DDD/444), and stores vector information in sets of three trytes (allowing for a total of 121 interconnected lines in 729 colors). They are drawn in succession, and black lines are not painted (there won't be black spots where black lines intersect visible ones.)

	MST					LST
First tryte	MRed	LRed	MGreen	LGreen	MBlue	LBlue
Second tryte	UNUSED	X	Coo	rdi	na	te
Third tryte	UNUSED	Y	Coo	rdi	na	te

Raster mode

Raster mode resolution is 324x243 (and uses a total of 108 memory pages), with one pixel per tryte, allowing for a total of 729 colors. They are encoded like in the first tryte in vector mode above.

Text mode

Text mode resolution is 27 rows by 54 columns (2 memory pages).

Interrupts

IRQ	Function	IRQ data
0	Keypress interrupt	Key
1	Clock interrupt	Random value
2	Keyboard break sent	undef.
3	Arithmetic error	undef.
4	Soft interrupt	undef.
5	Mouse motion	relative X : relative Y
6	Mouse event	click = 1, release = -1

Disk I/O

Tunguska features a floppydisk-like virtual disk drive. It is controlled through the Disk I/O tryte (DDD:DDA). It is the same size as the main memory (729*729 trytes). All data transfer is on the block level, and block size is 729 trytes. The following operations are allowed:

Name	Number	Function
NOOP	0	Idle
READ	1	Read from disk to memory
WRITE	2	Write from memory to disk
SYNC	3	Write from virtual disk to physical file
SEEK	4	Set disk position to page
GETPOS	5	Get disk position

SEEK and GETPOS use the accumulator as input. Data is read to and from memory page specified by the Y register.

Logic tables

A	**B**	**A$\wedge$B**	**A$\oplus$B**	**A BUT B**	**A$\vee$B**	**A TSH B**	**A PRM B**
+	+	+	-	+	+	+	-
+	0	0	0	0	+	+	+
+	-	-	+	-	+	0	0
0	+	0	0	0	+	+	+
0	0	0	0	0	0	0	0
0	-	-	0	0	0	-	-
-	+	-	+	-	+	0	0
-	0	-	0	0	0	-	-
-	-	-	+	-	-	-	+

Notes for 6502 programmers

There are some major differences between the 6502 and this machine. Beyond the obvious differences in endianness, and processor status flag, some instructions have changed. **PHA** is replaced by a general purpose **PSH** ("*PHA A*" will replicate the behavior of **PHA**), that will push any memory value onto the stack. **PLL** is it's respective pull operation.

X:Y addressing is a new mode that uses both the X and Y register to address a memory location. The **LAD** operator is a quick way to transfer a complex memory location onto X:Y (compare with the x86 **"LEA"** operation).