Instructions of DIY Calculator		
Control	NOP	No-operation, CPU doesn’t do anything.
	HALT	Generate internal NOPs until an interrupt occurs.
	SETIM	Set the interrupt mask flag in the status register.
	CLRIM	Clear the interrupt mask flag in the status register.
Arithmetic	ADD	Add data in memory to the accumulator.
	ADDC	Like an ADD, but include contents of the carry flag.
	SUB	Subtract data in memory from the accumulator.
	SUBC	Like a SUB, but include contents of the carry flag.
Logical	AND	AND data in memory to the accumulator.
	OR	OR data in memory to the accumulator.
	XOR	XOR data in memory to the accumulator.
Comparison, Shifts, and Rotates	CMPA	Compare data in memory to the accumulator.
	SHL	Shift the accumulator left 1 bit (arithmetic shift).
	SHR	Shift the accumulator right 1 bit (arithmetic shift).
	ROLC	Rotate the accumulator left 1 bit (through carry flag).
	RORC	Rotate the accumulator right 1 bit (through carry flag).
Increments and Decrements	INCA	Increment the accumulator.
	DECA	Decrement the accumulator.
	INCX	Increment the index register.
	DECX	Decrement the index register.
Loads and Stores	LDA	Load data in memory into the accumulator.
	STA	Store data in the accumulator into memory.
	BLDX	Load data in memory into the index register.
	BSTX	Store data in the index register into memory.
	BLDSP	Load data in memory into the stack pointer.
	BSTSP	Store data in the stack pointer into memory.
	BLDIV	Load data in memory into the interrupt vector.
Push and Pop	PUSHA	Push the accumulator onto the stack.
	POPA	Pop the accumulator from the stack.
	PUSHSR	Push the status register onto the stack.
	POPSR	Pop the status register from the stack.
Jumps	JMP	Jump to a new memory location.
	JSR	Jump to a subroutine.
	JZ	Jump if the result was zero.
	JNZ	Jump if the result wasn’t zero.
	JN	Jump if the result was negative.
	JNN	Jump if the result wasn’t negative.
	JC	Jump if the result generated a carry.
	JNC	Jump if the result didn’t generate a carry.
	JO	Jump if the result generated an overflow.
	JNO	Jump if the result didn’t generate an overflow.
Returns	RTS	Return from a subroutine.
	RTI	Return from an interrupt.

Character codes of DIY Calculator

hex	dec	char
$00	0	0
$01	1	1
$02	2	2
$03	3	3
$04	4	4
$05	5	5
$06	6	6
$07	7	7
$08	8	8
$09	9	9
$0A	10	A
$0B	11	B
$0C	12	C
$0D	13	D
$0E	14	E
$0F	15	F
$10	16	Clr
$11	17	Bell
$12	18	Back
$13	19	
$14	20	
$15	21	
$16	22	
$17	23	
$18	24	
$19	25	
$1A	26	
$1B	27	
$1C	28	
$1D	29	
$1E	30	
$1F	31	
$20	32	Space
$21	33	!
$22	34	"
$23	35	#
$24	36	$
$25	37	%
$26	38	&
$27	39	'
$28	40	(
$29	41	)
$2A	42	*
$2B	43	+
$2C	44	,
$2D	45	-
$2E	46	.
$2F	47	/
$30	48	0
$31	49	1
$32	50	2
$33	51	3
$34	52	4
$35	53	5
$36	54	6
$37	55	7
$38	56	8
$39	57	9
$3A	58	:
$3B	59	;
$3C	60	<
$3D	61	=
$3E	62	>
$3F	63	?
$40	64	@
$41	65	A
$42	66	B
$43	67	C
$44	68	D
$45	69	E
$46	70	F
$47	71	G
$48	72	H
$49	73	I
$4A	74	J
$4B	75	K
$4C	76	L
$4D	77	M
$4E	78	N
$4F	79	O
$50	80	P
$51	81	Q
$52	82	R
$53	83	S
$54	84	T
$55	85	U
$56	86	V
$57	87	W
$58	88	X
$59	89	Y
$5A	90	Z
$5B	91	[
$5C	92	\
$5D	93	]
$5E	94	^
$5F	95	_
$60	96	`
$61	97	a
$62	98	b
$63	99	c
$64	100	d
$65	101	e
$66	102	f
$67	103	g
$68	104	h
$69	105	i
$6A	106	j
$6B	107	k
$6C	108	l
$6D	109	m
$6E	110	n
$6F	111	o
$70	112	p
$71	113	q
$72	114	r
$73	115	s
$74	116	t
$75	117	u
$76	118	v
$77	119	w
$78	120	x
$79	121	y
$7A	122	z
$7B	123	{
$7C	124	|
$7D	125	}
$7E	126	~
$7F	127	

I/O ports			
Inputs		Outputs	
$F000	8-bit switch bank #1	$F020	8-bit LED display on workbench
$F001	8-bit switch bank #2	$F021	single undecoded 7-segment display
$F008	qwerty keyboard	$F022	single decoded 7-segment display
$F011	front panel keyboard	$F023	dual decoded 7-segment display
		$F028	console
		$F031	LCD display
		$F032	front panel LED display

Front panel button codes			
$10	Clear	$20	F-S
$11	CE	$21	Exp
$12	Back	$36	n!
$13	Enter	$37	Log
$14	+/–	$38	Tan
$15	.	$39	Cos
$16	+	$3A	Sin
$17	–	$3B	1/x
$18	*	$3C	Rx
$19	/	$3D	x^2
$1A	=	$3E	x^3
$1B	(	$3F	x^y
$1F	)	$40	Hex
$1C	Pi	$41	Dec
$1D	Mod	$42	Bin

Undecoded 7-segment display		
segment	hex	dec
upper segment	$01	%00000001
upper right segment	$02	%00000010
lower right segment	$04	%00000100
lower segment	$08	%00001000
lower left segment	$10	%00010000
upper left segment	$20	%00100000
middle segment	$40	%01000000