SS32 MISC

Six Shooter 32bits Minimal Instruction Set Computer

DSCF .co.uk

13/02/2025
VER 1.0(SS32X - Sheryl)
Notes: single core, single cycle execution, 32bit word addressed.
C only version runs on windows/linux, slower than PC32
FPGA version rated for 130mhz on Artix 100T-2C

Instruction Encoding

The SS32 packs up to 6 5bit instructions called slots per 32bit memory word in the format:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	6	6	6	6	6	5	5	5	5	5	4	4	4	4	4	3	3	3	3	3	2	2	2	2	2	1	1	1	1	1

Slots marked as REL use the sign extended remainder of the word after its slot in the format:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	I	I	I	I	I

This format can be used in any slot with the sign extended remainder **ROTW**(Rest Of The Word) ranging from 2^5 to 2^20 signed range, in the case of jumps 0 can also be a valid offset meaning 'this 32bit word'

Calls are encoded using 30bits direct address giving an effective range of 1GB available memory space in the format:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	0	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A

Instruction Execution

The SS32 architecture has 7 execution slots available with slot 0 being set aside for instruction fetching and/or interrupt servicing in the format:

06	05	04	03	02	01	00
exe	exe	exe	exe	exe	exe	Ifetch

If interrupt servicing is to be omitted the slot 0 instruction fetch can be ommitted as well so long as slot 6 is not a memory fetch.

Stacks and Registers

SS32 has 2 internal stacks:

Data Stack	16 deep, used for parameter passing and manipulation
Return Stack	32 deep, stores return addresses and serves as auxiliary storage for Data Stack

SS32 counts with internal registers both hidden and available to the user:

P	Program Counter, word addressed[hidden]
X	Instruction to eXecute[hidden]
T	Top of Data Stack
S	Second of Data Stack[hidden]
I	Top of Return Stack
A	Memory Read Address Register, variable addressing
B	Memory Write Address Register, variable addressing

Memory Layout

0x00000000-0x0000FFEF	256KB 64KW SRAM0
0x0000FF00-0x0000FFFF	Internal registers(RESET, DMA, IO, Video)
0x00010000-0x0001FFFF	256KB 64KW SRAM1
0x00020000-0x000FFFFF	4MB 1MW Battery Backed Storage
0x00100000-0x00FFFFFF	60MB 15MW DRAM

*KB=1024bytes *KW=1024words(32bit) *MB=1024*1024bytes *MW=1024*1024words(32bit)

Instruction Set

[00] NOP	Terminates current word and fetches next word
[01] RET	Pops R into P, fetches next word
[02] JMP	Adds ROTW to P, fetches next word
[03] JZ	JMPs if T is 0 or if T's sign bit is set
[04] J1	JMPs if T>0
[05] NEXT	JMPs if R>0, decreases R by 1
[06] SHL	Shifts T left by ROTW, max 31. Fetches next word
[07] SHR	Shifts T right by ROTW, if ROTW is signed shifts arithmetically. Fetches next word
[08] SMD	Changes Instruction set with ROTW as the index. Fetches next word
[09] LIT	Pushes the contents of P into Data Stack, adds 1 to P
[10] EXE	Pushes P+1 into Return Stack, drops T into P. Fetches next word
[11] PUSH	Pushes T into Return Stack
[12] POP	Pops Return Stack into T
[13] STORE	Stores S into address T, drops S
[14] FETCH	Fetches address T into T
[15] IREG	Pushes I into Data Stack
[16] AREG	Pushes Register A into Data Stack
[17] BREG	Pushes Register B into Data Stack
[18] STOA	Drops T into Register A
[19] STOB	Drops T into Register B
[20] FAP	Fetch A Plus, fetches A into T then adds 1 to A
[21] SBP	Drops T into address B then adds 1 to B
[22] DUP	Pushes T into Data Stack(dup)
[23] DRP	Pops Data Stack into T(drop)
[24] SWP	Swaps S and T
[25] OVR	Pushes T into Data Stack, moves S into T
[26] ADD	Adds T and S, discards S
[27] AND	Logical AND T and S, discards S
[28] EOR	Exclussive OR T and S, discards S
[29] NOT	Logical NOT T
[30] NEG	Negates T(T*-1)
[31] MUL	Multiplies T and S, discards S

Multiple Instruction Sets

The **SMD** instruction enables change to arbitrary instruction sets, currently:

0	32bit word addressed memory accesses(default)
1	128bit SIMD Vector operation mode TBD
2	32bit GPU operation mode TBD
3	8bit byte addressed memory accesses