VMS CPU

The CPU in the VMS is a customized Sanyo microcontroller which is code compatible with the LC86000 series.

For assembler and other development tools, check out the software page.

Memory model

The CPU uses two disjoint address spaces, ROM space which is used for fetching instructions and by the LDC instruction, and RAM space which is used for loading and storing operands. ROM space is 65536 bytes large, and on the VMS in game mode, it is the same as the first half of the flash memory. RAM space is 512 bytes large, and is divided in two halves. The first half (address 0 though FF) is general purpose RAM, used for variable storage. The second half (address 100 through 1FF) is used for special function registers (SFR area). The general purpose RAM area can be mapped to one of two banks of 256 bytes each, using the PSW register. The first of these banks is used by the VMS firmware, and contains system variables. The CPU stack is also located in this bank. The other bank is exclusively for application use, and selected by the firmware before game mode is entered.

A complete list of the special function registers is available.

Addressing modes

Many instructions operate implicitly on some fixed SFR(s) (typically ACC). Additional operands in either ROM or RAM space can be added using one or more of the following addressing modes:

Immediate (i8)
: An 8-bit integer constant immediately following the opcode in ROM space.

Direct (d9)
: Any location in RAM space (either RAM or an SFR) specified with an absolute 9-bit address.

Indirect (@Ri)
: A RAM or SFR location specified with the contents of a RAM cell (see below).

Bit specifier (b3)
: A 3-bit integer constant indicating which bit (0-7) to be affected by the instruction. 0 is the least significant bit, and 7 is the most significant bit.

Absolute (a12/a16)
: A program location in ROM space specified with an absolute 12- or 16-bit address. In the case of a 12-bit address, the four most significant bits of PC will remain unaffected. That mode is thus restricted to jumping within the same 4K segment. In both modes, the address is given in big endian byte order.

Relative (r8/r16)
: A program location in ROM space specified with a relative offset from the address of the following instruction. An 8-bit offset is treated as signed, whereas a 16-bit offset is treated as unsigned. 16-bit offsets are given in **little** endian byte order.

The indirect addressing mode requires some more detailed explanation. As said earlier, it uses the contents of a memory cell to determine the RAM/SFR address of the operand. There are four possible indirection modes,

@R0 through @R3, which (normally) uses the RAM locations 0 through 3 respecitively to determine the target address. Since these locations only contain 8 bits of data each, they don't span the entire RAM space. Instead, the most signigicant bit of the address is taken from the most significant bit of the indirection mode number. This means that @R0 and @R1 will always access the RAM half and that @R2 and @R3 will always access the SFR half.

To make things even more interresting, the memory cells used for fetching the target address aren't locked to address 0-3, but can be selected from four possible ranges. This selection is done with two bits in the PSW register, IRBK0 and IRBK1 (bit 3 and 4).

Summary:

IRBK1	IRBK0	Mode	Address read from	Operand in
0	0	@R0	000	RAM (000-0FF)
		@R1	001	RAM (000-0FF)
		@R2	002	SFR (100-1FF)
		@R3	003	SFR (100-1FF)
0	1	@R0	004	RAM (000-0FF)
		@R1	005	RAM (000-0FF)
		@R2	006	SFR (100-1FF)
		@R3	007	SFR (100-1FF)
1	0	@R0	008	RAM (000-0FF)
		@R1	009	RAM (000-0FF)
		@R2	00A	SFR (100-1FF)
		@R3	00B	SFR (100-1FF)
1	1	@R0	00C	RAM (000-0FF)
		@R1	00D	RAM (000-0FF)
		@R2	00E	SFR (100-1FF)
		@R3	00F	SFR (100-1FF)

Interrupt vectors

When an interrupt occurs, the CPU branches to a predetermined address as shown by the following table. Execution can be resumed at the previous location using the `RETI` instruction.

Address	Cause
00	Reset
03	INT0 interrupt (external)
0B	INT1 interrupt (external)
13	INT2 interrupt (external) or T0L overflow
1B	INT3 interrupt (external) or Base Timer overflow
23	T0H overflow

2B	T1H or T1L overflow
33	SIO0 interrupt
3B	SIO1 interrupt
43	RFB interrupt
4B	P3 interrupt

Instruction set

	0	1	2, 3	4-7	8-F
0	NOP	BR r8	LD d9	LD @Ri	CALL a12
1	CALLR r16	BRF r16	ST d9	ST @Ri	
2	CALLF a16	JMPF a16	MOV #i8,d9	MOV #i8,@Ri	JMP a12
3	MUL	BE #i8,r8	BE d9,r8	BE @Ri,#i8,r8	
4	DIV	BNE #i8,r8	BNE d9,r8	BNE @Ri,#i8,r8	BPC d9,b3,r8
5			DBNZ d9,r8	DBNZ @Ri,r8	
6	PUSH d9		INC d9	INC @Ri	BP d9,b3,r8
7	POP d9		DEC d9	DEC @Ri	
8	BZ r8	ADD #i8	ADD d9	ADD @Ri	BN d9,b3,r8
9	BNZ r8	ADDC #i8	ADDC d9	ADDC @Ri	
A	RET	SUB #i8	SUB d9	SUB @Ri	NOT1 d9,b3
B	RETI	SUBC #i8	SUBC d9	SUBC @Ri	
C	ROR	LDC	XCH d9	XCH @Ri	CLR1 d9,b3
D	RORC	OR #i8	OR d9	OR @Ri	
E	ROL	AND #i8	AND d9	AND @Ri	SET1 d9,b3
F	ROLC	XOR #i8	XOR d9	XOR @Ri	

ADD

Mnemonic	Code	Cycles
ADD #i8	10000001 i7i6i5i4i3i2i1i0	1
ADD d9	1000001d8 d7d6d5d4d3d2d1d0	1
ADD @Ri	100001i1i0	1

Add the operand to the ACC register. CY, AC and OV are set according to the result.

ADDC

Mnemonic	Code	Cycles
ADDC #i8	10010001 i7i6i5i4i3i2i1i0	1

ADDC d9	1001001d8 d7d6d5d4d3d2d1d0	1
ADDC @Ri	100101i1i0	1

Add the operand and the carry bit (CY) to the ACC register. CY, AC and OV are set according to the result.

SUB

Mnemonic	Code	Cycles
SUB #i8	10100001 i7i6i5i4i3i2i1i0	1
SUB d9	1010001d8 d7d6d5d4d3d2d1d0	1
SUB @Ri	101001i1i0	1

Subtract the operand from the ACC register. CY, AC and OV are set according to the result.

SUBC

Mnemonic	Code	Cycles
SUBC #i8	10110001 i7i6i5i4i3i2i1i0	1
SUBC d9	1011001d8 d7d6d5d4d3d2d1d0	1
SUBC @Ri	101101i1i0	1

Subtract the operand and the carry bit (CY) from the ACC register. CY, AC and OV are set according to the result.

INC

Mnemonic	Code	Cycles
INC d9	0110001d8 d7d6d5d4d3d2d1d0	1
INC @Ri	011001i1i0	1

Increment the operand by one. No PSW flags are affected.

DEC

Mnemonic	Code	Cycles
DEC d9	0111001d8 d7d6d5d4d3d2d1d0	1
DEC @Ri	011101i1i0	1

Decrement the operand by one. No PSW flags are affected.

MUL

Mnemonic	Code	Cycles
MUL	00110000	7

Perform a multiplication. The ACC and C registers together form a 16-bit operand (ACC being the high 8 bits, and C being the low 8 bits) which is multiplied by the contents of the B register. The result is a 24-bit number that is stored in the ACC, C and B registers (the high 8 bits are stored in B, the middle 8 bits in ACC, and the

low 8 bits in C). CY is cleared, and OV is set if the result is greater than 16 bits, otherwise cleared. AC is not affected.

DIV

Mnemonic	Code	Cycles
DIV	01000000	7

Perform a division. The ACC and C registers together form a 16-bit operand (ACC being the high 8 bits, and C being the low 8 bits) which is divided by the contents of the B register. The result is a 16-bit quotient that is stored in ACC and C (the high 8 bits in ACC, and the low 8 bits in C), and an 8-bit remainder that is stored in B. CY is cleared, and OV is set if the remainder is zero, otherwise cleared. AC is not affected.

AND

Mnemonic	Code	Cycles
AND #i8	11100001 i7i6i5i4i3i2i1i0	1
AND d9	1110001d8 d7d6d5d4d3d2d1d0	1
AND @Ri	111001i1i0	1

Perform bitwise AND between the operand and the ACC register. No PSW flags are affected.

OR

Mnemonic	Code	Cycles
OR #i8	11010001 i7i6i5i4i3i2i1i0	1
OR d9	1101001d8 d7d6d5d4d3d2d1d0	1
OR @Ri	110101i1i0	1

Perform bitwise OR between the operand and the ACC register. No PSW flags are affected.

XOR

Mnemonic	Code	Cycles
XOR #i8	11110001 i7i6i5i4i3i2i1i0	1
XOR d9	1111001d8 d7d6d5d4d3d2d1d0	1
XOR @Ri	111101i1i0	1

Perform bitwise XOR between the operand and the ACC register. No PSW flags are affected.

ROL

Mnemonic	Code	Cycles
ROL	11100000	1

Rotate the contents of the ACC register one bit to the left. The most signigicant bit will wrap immediately around to the least signigicant bit. No PSW flags are affected.

ROLC

Mnemonic	Code	Cycles
ROLC	11110000	1

Rotate the contents of the ACC register one bit to the left. The most signigicant bit is copied to the CY flag, and the old value of CY will be place in the least signigicant bit. The AC and OV flags are unaffected.

ROR

Mnemonic	Code	Cycles
ROR	11000000	1

Rotate the contents of the ACC register one bit to the right. The least signigicant bit will wrap immediately around to the most signigicant bit. No PSW flags are affected.

RORC

Mnemonic	Code	Cycles
RORC	11010000	1

Rotate the contents of the ACC register one bit to the right. The least signigicant bit is copied to the CY flag, and the old value of CY will be place in the most signigicant bit. The AC and OV flags are unaffected.

LD

Mnemonic	Code	Cycles
LD d9	0000001d8 d7d6d5d4d3d2d1d0	1
LD @Ri	000001i1i0	1

Load the operand into the ACC register. No PSW flags are affected.

ST

Mnemonic	Code	Cycles
ST d9	0001001d8 d7d6d5d4d3d2d1d0	1
ST @Ri	000101i1i0	1

Store the contents of the ACC register into the operand address. No PSW flags are affected.

MOV

Mnemonic	Code	Cycles
MOV #i8,d9	0010001d8 d7d6d5d4d3d2d1d0 i7i6i5i4i3i2i1i0	2
MOV #i8,@Rj	001001j1j0 i7i6i5i4i3i2i1i0	1

Set the contents of the operand to a constant value. No PSW flags are affected.

LDC

Mnemonic	Code	Cycles
LDC	11000001	2

Load a constant from ROM space into the ACC register. The ROM address is formed by adding the old value of ACC to the contents of the TRH and TRL registers viewed as a 16-bit value (TRH being the upper 8 bits, and TRL being the lower 8 bits). No PSW flags are affected.

PUSH

Mnemonic	Code	Cycles
PUSH d9	0110000d8 d7d6d5d4d3d2d1d0	2

Push the operand on the stack. The SP register is first incremented by one, and the operand value is then stored at the resulting stack position. No PSW flags are affected.

POP

Mnemonic	Code	Cycles
POP d9	0111000d8 d7d6d5d4d3d2d1d0	2

Pop the operand from the stack. The value is read from the stack position pointed out by the current value of the SP register, and SP is then decremented by one. No PSW flags are affected.

XCH

Mnemonic	Code	Cycles
XCH d9	1100001d8 d7d6d5d4d3d2d1d0	1
XCH @Ri	110001i1i0	1

Exchange the contents of the operand with the contents of the ACC register. No PSW flags are affected.

JMP

Mnemonic	Code	Cycles
JMP a12	001a111a10a9a8 a7a6a5a4a3a2a1a0	2

Jump unconditionally. The target address is specified using a 12-bit absolute address, so the upper 4 bits of this address must be the same as for the instruction following the JMP. No PSW flags are affected.

JMPF

Mnemonic	Code	Cycles
JMPF a16	00100001 a15a14a13a12a11a10a9a8 a7a6a5a4a3a2a1a0	2

Jump unconditionally. The target address is specified using a full 16-bit absolute address. No PSW flags are affected.

BR

Mnemonic	Code	Cycles

BR r8	00000001 r7r6r5r4r3r2r1r0	2

Branch unconditionally. The target address is specified using an 8-bit relative address. The signed 8-bit offset is added to the address of the instruction following the BR. No PSW flags are affected.

BRF

Mnemonic	Code	Cycles
BRF r16	00010001 r7r6r5r4r3r2r1r0 r15r14r13r12r11r10r9r8	4

Branch unconditionally. The target address is specified using a 16-bit relative address. The unsigned 16-bit offset is added to the address of the instruction following the BRF minus one to produce the target address. The addition is performed modulo 65536, which makes it possible to branch to a lower address as well. No PSW flags are affected.

BZ

Mnemonic	Code	Cycles
BZ r8	10000000 r7r6r5r4r3r2r1r0	2

Branch if the ACC register is zero. See BR for address calculation. No PSW flags are affected.

BNZ

Mnemonic	Code	Cycles
BNZ r8	10010000 r7r6r5r4r3r2r1r0	2

Branch if the ACC register is not zero. See BR for address calculation. No PSW flags are affected.

BP

Mnemonic	Code	Cycles
BP d9,b3,r8	011d81b2b1b0 d7d6d5d4d3d2d1d0 r7r6r5r4r3r2r1r0	2

Branch if the specified bit of the operand is set. See BR for address calculation. No PSW flags are affected.

BPC

Mnemonic	Code	Cycles
BPC d9,b3,r8	010d81b2b1b0 d7d6d5d4d3d2d1d0 r7r6r5r4r3r2r1r0	2

If the specified bit of the operand is set, clear the bit and branch. See BR for address calculation. No PSW flags are affected.

BN

Mnemonic	Code	Cycles
BN d9,b3,r8	100d81b2b1b0 d7d6d5d4d3d2d1d0 r7r6r5r4r3r2r1r0	2

Branch if the specified bit of the operand is not set. See BR for address calculation. No PSW flags are affected.

DBNZ

Mnemonic	Code	Cycles
DBNZ d9,r8	0101001d8 d7d6d5d4d3d2d1d0 r7r6r5r4r3r2r1r0	2
DBNZ @Ri,r8	010101i1i0 r7r6r5r4r3r2r1r0	2

Decrement the operand by one, and branch if the result is not zero. See BR for address calculation. No PSW flags are affected.

BE

Mnemonic	Code	Cycles
BE #i8,r8	00110001 i7i6i5i4i3i2i1i0 r7r6r5r4r3r2r1r0	2
BE d9,r8	0011001d8 d7d6d5d4d3d2d1d0 r7r6r5r4r3r2r1r0	2
BE @Rj,#i8,r8	001101j1j0 i7i6i5i4i3i2i1i0 r7r6r5r4r3r2r1r0	2

Branch if the contents of the ACC register (or the indirect operand in the third form above) are equal to the immediate or direct operand. See BR for address calculation. Additionally, CY is set to 1 if ACC (or the indirect operand) is strictly less than the immediate or direct operand. AC and OV are unaffected.

BNE

Mnemonic	Code	Cycles
BNE #i8,r8	01000001 i7i6i5i4i3i2i1i0 r7r6r5r4r3r2r1r0	2
BNE d9,r8	0100001d8 d7d6d5d4d3d2d1d0 r7r6r5r4r3r2r1r0	2
BNE @Rj,#i8,r8	010001j1j0 i7i6i5i4i3i2i1i0 r7r6r5r4r3r2r1r0	2

Branch if the contents of the ACC register (or the indirect operand in the third form above) are not equal to the immediate or direct operand. See BR for address calculation. Additionally, CY is set to 1 if ACC (or the indirect operand) is strictly less than the immediate or direct operand. AC and OV are unaffected.

CALL

Mnemonic	Code	Cycles
CALL a12	000a111a10a9a8 a7a6a5a4a3a2a1a0	2

Call function. The entry address of the function is specified using a 12-bit absolute address, so the upper 4 bits of this address must be the same as for the instruction following the CALL. The return address (the address of the instruction following the CALL instruction) is pushed on the stack. The lower 8 bits of the address are pushed first, then the upper 8 bits. No PSW flags are affected.

CALLF

Mnemonic	Code	Cycles
CALLF a16	00100000 a15a14a13a12a11a10a9a8 a7a6a5a4a3a2a1a0	2

Call function. The entry address of the function is specified using a full 16-bit absolute address. The return address (the address of the instruction following the CALLF instruction) is pushed on the stack. The lower 8 bits of the address are pushed first, then the upper 8 bits. No PSW flags are affected.

CALLR

Mnemonic	Code	Cycles
CALLR r16	00010000 r7r6r5r4r3r2r1r0 r15r14r13r12r11r10r9r8	4

Call function. The entry address of the function is specified using a 16-bit relative address. The unsigned 16-bit offset is added to the address of the instruction following the CALLR minus one to produce the target address. The addition is performed modulo 65536, which makes it possible to call a lower address as well. The return address (the address of the instruction following the CALLR instruction) is pushed on the stack. The lower 8 bits of the address are pushed first, then the upper 8 bits. No PSW flags are affected.

RET

Mnemonic	Code	Cycles
RET	10100000	2

Return from function. The PC register is popped from the stack. The upper 8 bits are popped first, then the lower 8 bits. No PSW flags are affected.

RETI

Mnemonic	Code	Cycles
RETI	10110000	2

Return from interrupt. The PC register is popped from the stack. The upper 8 bits are popped first, then the lower 8 bits. No PSW flags are affected.

CLR1

Mnemonic	Code	Cycles
CLR1 d9,b3	110d81b2b1b0 d7d6d5d4d3d2d1d0	1

Clear the specified bit in the operand. No PSW flags are affected.

SET1

Mnemonic	Code	Cycles
SET1 d9,b3	111d81b2b1b0 d7d6d5d4d3d2d1d0	1

Set the specified bit in the operand. No PSW flags are affected.

NOT1

Mnemonic	Code	Cycles
NOT1 d9,b3	101d81b2b1b0 d7d6d5d4d3d2d1d0	1

Invert the specified bit in the operand. No PSW flags are affected.

NOP

Mnemonic	Code	Cycles
NOP	00000000	1

Do nothing. No PSW flags are affected.

Dreamcast Programming by *Marcus Comstedt*
Last modified: Mon Mar 27 17:27:05 MEST 2000