Summary

The EC16 is a microprocessor core written in VHDL with the following features

- 64K * 16-bit external memory space for code, data and I/O
- 256 * 16-bit internal ram for registers, pointers, stack and scratchpad
- 50 instructions
  - length: 46 one-word, 4 two-word
  - speed: 18 one-cycle, 15 two-cycle, 9 three-cycle, 8 one/two-cycle (branch)
- 4 maskable prioritized interrupts

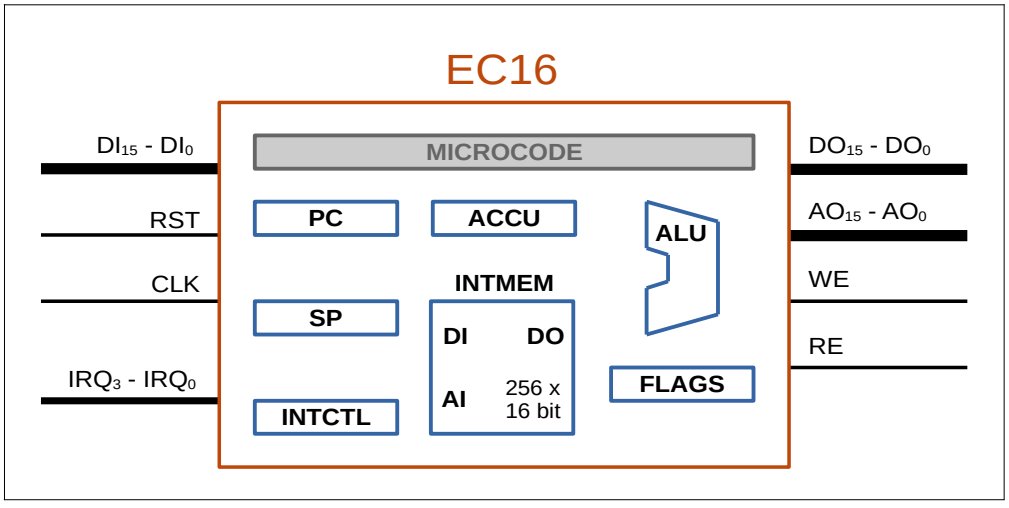


Figure 1: simplified block diagram

Memory Model

The EC16 has two completely separate memory spaces, the external memory space (EXTMEM) and a small but faster accessible internal memory (INTMEM).

While the EXTMEM space can be populated as required with ram, rom and I/O devices, the INTMEM space is always fully populated with an EBRam organized as 256 x 16 bit.

The INTMEM cells can be used as registers, pointers to INTMEM or EXTMEM locations, scratchpad ram or stack space. The stack pointer is a separate register that is initialized to 0xFF at reset and grows towards 0x00. The usual caution has to be exercised to avoid a conflict between the growing stack and the other variables.

Internal Memory (INTMEM) 256 x 16 bit				
Registers Pointers Sratchpad	⇳	0xFF 0x00	⇩	Stack

External Memory (EXTMEM) 64K x 16 bit	
0xFFFF ⇳	Code, Data, I/O
0x0020	IRQ3
0x0018	IRQ2
0x0010	IRQ1
0x0008	IRQ0
0x0000	Reset

The EXTMEM must be populated with at least a small amount of ROM (pre-initialized RAM) beginning at address 0 since the reset and interrupt vectors are located there. The rest of the address space can be used as required.

The EC16 has a reduced instruction set architecture that is strongly focussed on the use of the faster INTMEM. Each of the 256 INTMEM locations can be used as a register or a pointer and its 8-bit address is encoded in the lower half of the opcode which speeds up fetch- and execution time. Read and write access to the EXTMEM space is handled by

only two instructions (MOVXI A INTMEM / MOVXI INTMEM A), both of which use an INTMEM cell to provide the address (indirect addressing via pointer). Nevertheless block operations can be very efficiently implemented with only three registers (source, destination and counter), in combination with the INC/DEC- and branch instructions.

Interrupts

The EC16 has four interrupt inputs : IRQ0 – IRQ3. IRQ0 has the highest, IRQ3 the lowest priority. All four interupts are triggered by a rising edge. Prioritization and execution are handled via the INTCTL block. This block contains i.a. three registers : IRR (Interrupt Request Register), IMASK and IIP (Interrupt In Progress).

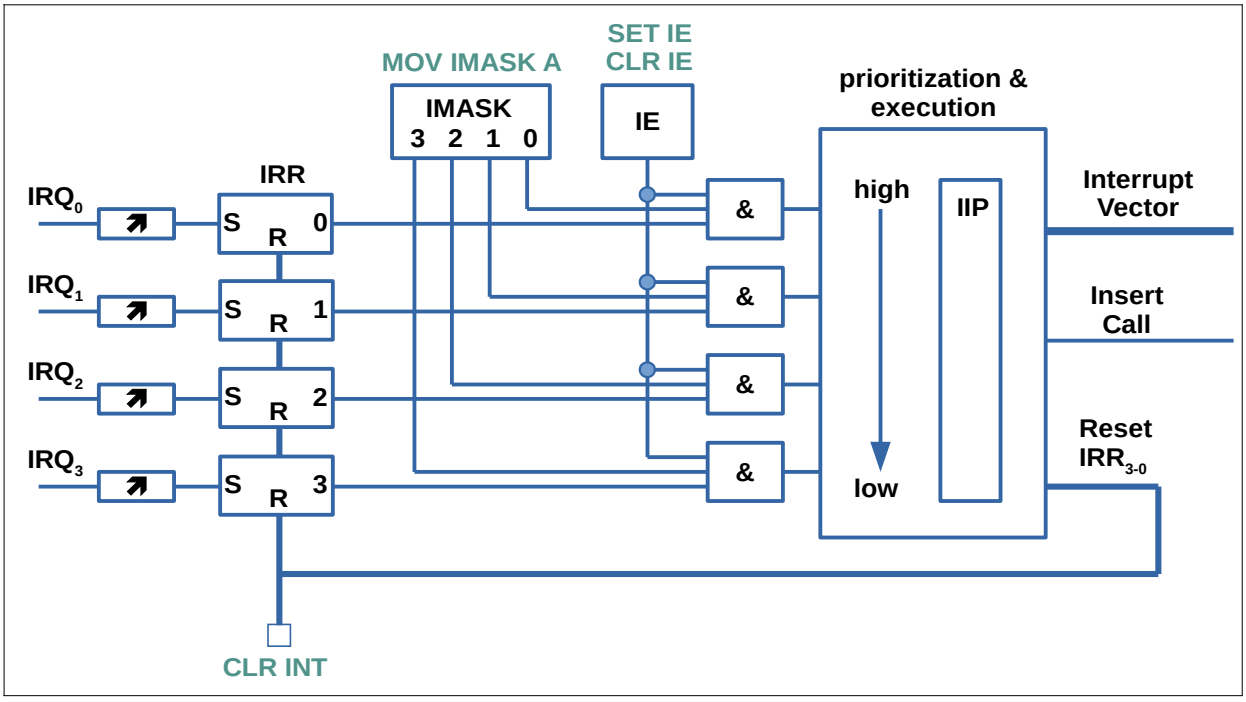


Figure 2: INTCTL (Interrupt Control Block)

A rising edge on IRQ_n is always registered in IRR_n regardless of the state of $IMASK_n$ and IE. If, however, the rising edge occurs in the same clock cycle as a CLR INT instruction (which resets all IRR flags at once), the clear instruction takes precedence.

$IMASK_n$ enables (1) or disables (0) IRQ_n. IMASK can be written with a MOV IMASK A instruction, which transfers the four least significant bits of the AKKU to $IMASK_{3-0}$.

The flag IE enables (1) or disables (0) all interrupts simultaneously. IE can be set/reset with the SET IE, CLR IE and the MOV FLAGS A instructions.

Unless it is cleared by a CLR INT instruction, the IRR_n flag remains set until its interrupt is executed. Interrupt execution happens as soon as IRR_n, $IMASK_n$ and IE are all set and no interrupt with higher priority is in progress.

The current instruction is finished, IRR_n is cleared and IIP_n is set instead, blocking all lower level interrupts. Then the address of the next instruction is pushed onto the stack and a call to the corresponding interrupt vector is executed. IIP_n stays set until the interrupt service routine is finished with a RETI instruction. RETI clears IIP_n, pops the return address from the stack and resumes executing from where it left of.

Notice that as soon as IRR_n is cleared, it will register the next rising edge on IRQ_n, even if the current ISR (Interrupt Service Routine) is still in progress.

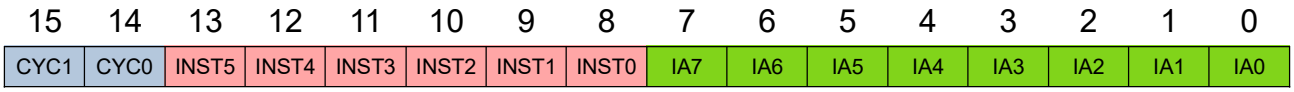
Notice furthermore that at least one instruction of the interrupted program context is executed before the next interrupt is taken.

An IRQ with higher priority will interrupt the current ISR. This can be avoided if necessary by temporarily clearing the IE flag. Typical examples are atomic operations like a read-modify-write access to a peripheral or the handling of semaphores. Only the crucial instructions should be framed by a pair of CLR IE and SET IE.

Each interrupt level needs one word of stack space for the return address. No variables, registers or flags are saved automatically. It is up to the user to take care of it.

Instruction Set

The EC16 opcodes consist of three fields



- **Bit 15:14** - CYC1:0 : number of clock cycles

CYC1:0	Number of clock cycles
00	1
01	2
10	3
11	2/1 (branch/no branch)

- **Bit 13:8** - INST5:0 : instruction index
- **Bit 7:0** - IA7:0 : depending on instruction :
  - INTMEM address (8 bit unsigned)
  - Branch offset (8 bit signed)
  - 0 if none of the above applies

The instructions JMPD U16, CALLD U16, MOVD A U16 and MOVD INTMEM U16 are two-word instructions, i.e. the opcode is followed by a 16 bit constant. In case of JMPD and CALLD this is the target address, in case of MOVD the value is loaded into the Accu respectively the INTMEM cell.

Instructions sorted by Mnemonic			
Mnemonic	OPCode	Cycles	Words
ADD A INTMEM	43u8	2	1
ADDC A INTMEM	41u8	2	1
AND A INTMEM	4Cu8	2	1
BRCC S8	C0s8	2/1	1
BRCS S8	C4s8	2/1	1
BRNC S8	C2s8	2/1	1
BRNS S8	C6s8	2/1	1
BROC S8	C1s8	2/1	1
BROS S8	C5s8	2/1	1
BRZC S8	C3s8	2/1	1
BRZS S8	C7s8	2/1	1
CALLD U16	A100	3	2
CALLI INTMEM	A3u8	3	1
CLR C	1000	1	1
CLR IE	0200	1	1
CLR INT	0400	1	1
CMP A INTMEM	44u8	2	1
DEC INTMEM	46u8	2	1
INC INTMEM	47u8	2	1
JMPD U16	A000	3	2
JMPI INTMEM	A2u8	3	1
MOV A INTMEM	50u8	2	1
MOV A STATUS	1400	1	1
MOV FLAGS A	1200	1	1
MOV IMASK A	0500	1	1
MOV INTMEM A	16u8	1	1
MOV SP A	1300	1	1
MOVD A U16	6000	2	2
MOVD INTMEM U16	61u8	2	2
MOVI A INTMEM	80u8	3	1
MOVI INTMEM A	52u8	2	1
MOVXI A INTMEM	83u8	3	1
MOVXI INTMEM A	82u8	3	1
NOP	0000	1	1
NOT A	2F00	1	1
OR A INTMEM	4Du8	2	1
POP A	5100	2	1
PUSH A	1500	1	1
RETI	8500	3	1
RETS	8400	3	1
ROL A	2800	1	1
ROR A	2900	1	1
SET C	1100	1	1
SET IE	0300	1	1

SHL A	2A00	1	1
SHR A	2B00	1	1
SUB A INTMEM	42u8	2	1
SUBB A INTMEM	40u8	2	1
SWAP A	2500	1	1
XOR A INTMEM	4Eu8	2	1

Instructions sorted by OPCode			
Mnemonic	OPCode	Cycles	Words
NOP	0000	1	1
CLR IE	0200	1	1
SET IE	0300	1	1
CLR INT	0400	1	1
MOV IMASK A	0500	1	1
CLR C	1000	1	1
SET C	1100	1	1
MOV FLAGS A	1200	1	1
MOV SP A	1300	1	1
MOV A STATUS	1400	1	1
PUSH A	1500	1	1
MOV INTMEM A	16u8	1	1
SWAP A	2500	1	1
ROL A	2800	1	1
ROR A	2900	1	1
SHL A	2A00	1	1
SHR A	2B00	1	1
NOT A	2F00	1	1
SUBB A INTMEM	40u8	2	1
ADDC A INTMEM	41u8	2	1
SUB A INTMEM	42u8	2	1
ADD A INTMEM	43u8	2	1
CMP A INTMEM	44u8	2	1
DEC INTMEM	46u8	2	1
INC INTMEM	47u8	2	1
AND A INTMEM	4Cu8	2	1
OR A INTMEM	4Du8	2	1
XOR A INTMEM	4Eu8	2	1
MOV A INTMEM	50u8	2	1
POP A	5100	2	1
MOVI INTMEM A	52u8	2	1
MOVD A U16	6000	2	2
MOVD INTMEM U16	61u8	2	2
MOVI A INTMEM	80u8	3	1
MOVXI INTMEM A	82u8	3	1
MOVXI A INTMEM	83u8	3	1
RETS	8400	3	1

RETI	8500	3	1
JMPD U16	A000	3	2
CALLD U16	A100	3	2
JMPI INTMEM	A2u8	3	1
CALLI INTMEM	A3u8	3	1
BRCC S8	C0s8	2/1	1
BROC S8	C1s8	2/1	1
BRNC S8	C2s8	2/1	1
BRZC S8	C3s8	2/1	1
BRCS S8	C4s8	2/1	1
BROS S8	C5s8	2/1	1
BRNS S8	C6s8	2/1	1
BRZS S8	C7s8	2/1	1

ADD A INTMEM

OP-Code	Arg	Words	Cycles
43*u8*	-	1	2

Function:

ACCU ← ACCU + INTMEM(u8);

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

Add ACCU and INTMEM.

The contents of ACCU and INTMEM (addressed by the lower 8 bits of the OP-Code) are added and the result is stored into the ACCU.

The flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set when sign overflows, otherwise cleared
- CARRY set when result is bigger than 16 bits, otherwise cleared

Remarks:

This instruction does **not** add the current CARRY to the two summands. It can be used for a 16-bit addition or to add the lowest two words in a higher precision (e.g. 64-bit) addition.

ADDC A INTMEM

OP-Code	Arg	Words	Cycles
41*u8*	-	1	2

Function:

ACCU ← ACCU + INTMEM(u8) + CARRY;

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

ADD CARRY, ACCU and INTMEM.

The contents of ACCU, INTMEM (addressed by the lower 8 bits of the OP-Code) and CARRY flag are added and the result is stored into ACCU.

The flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set when sign overflows, otherwise cleared
- CARRY set when result is bigger than 16 bits, otherwise cleared

Remarks:

ADDC is normally used in higher precision (e.g. 64-bit) additions. At first the lowest two words are added with the ADD instruction. Then the higher words are added with the ADDC instruction, taking the carries into account.

AND A INTMEM

OP-Code	Arg	Words	Cycles
4C*u8*	-	1	2

Function:

ACCU ← ACCU **AND** INTMEM(u8)

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER

Description:

And ACCU and INTMEM.

The contents of ACCU and INTMEM (addressed by the lower 8 bits of the OP-Code) are ANDed bit-wise and the result is stored into the ACCU.

The flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set to zero

Remarks:

none

BRxx

OP-Code	Arg	Words	Cycles
Cx*s8*	-	1	2/1

Function:

Conditional branch

If condition is false then PC ← PC + 1

If condition is true then PC ← PC + s8 (branch)

Description:

Branch on (flag) clear / set.

If the condition is not met, the program simply continues with the next instruction. This takes one cycle.

If the condition is met, the offset (sign extended lower half of the OP-Code) is added to the program counter (PC). At this moment the PC is already pointing to the instruction following the branch instruction. So an offset of 0 will act as if no branch was taken. An offset of -1 will set the PC back to the branch instruction itself, thus creating an infinite loop. A branch reaches from -127 to +128 of its own address and takes two cycles.

x = branch taken

FLAG	CARRY		NEG		OVER		ZERO	
Mnemonic (Opcode)	0	1	0	1	0	1	0	1
BRCC (C0s8)	x							
BRCS (C4s8)		x						
BRNC (C2s8)			x					
BRNS (C6s8)				x				
BROC (C1s8)					x			
BROS (C5s8)						x		
BRZC (C3s8)							x	
BRZS (C7s8)								x

Remarks: none

CALLD U16

OP-Code	Arg	Words	Cycles
A100	#U16	2	3

Function:

INTMEM(SP) ← PC

PC ← #U16

Affected Flags: none

Description:

Call direct address.

The call address is the argument, i.e. the word following the opcode. First the return address, which is the address of the instruction following the argument, is pushed onto the stack. Then the PC is loaded with the call address and execution continues there.

Remarks:

One word of stack space must be available

CALLI INTMEM

OP-Code	Arg	Words	Cycles
A3*u8*	-	1	3

Function:

INTMEM(SP) ← PC+1

PC ← INTMEM(u8)

Affected Flags: none

Description:

Call indirect via pointer in INTMEM.

The call address is in INTMEM (which is addressed by the lower 8 bits of the OP-Code). First the return address, which is the address of the instruction following the CALL, is pushed onto the stack. Then the PC is loaded with the call address and execution continues there.

Remarks:

One word of stack space must be available

CLR C

OP-Code	Arg	Words	Cycles
1000	-	1	1

Function:

CARRY ← 0

PC ← PC + 1

Affected Flags: Carry

Description:

Clear CARRY.

The CARRY flag is set to zero. This can be used to prepare shift and rotate instructions.

Remarks:

none

CLR IE

OP-Code	Arg	Words	Cycles
0200	-	1	1

Function:

IE ← 0

PC ← PC + 1

Affected Flags: IE

Description:

Clear Interrupt Enable flag.

The Interrupt Enable flag is set to zero. Interrupts already in progress will not be affected.

Pending interrupts will stay pending but will not be taken until IE is set to one.

Remarks:

none

CLR INT

OP-Code	Arg	Words	Cycles
0400	-	1	1

Function:

IRRx ← 0

PC ← PC + 1

Affected Flags: none

Description:

Clear Interrupts.

All Interrupt Request Registers that hold pending interrupts are cleared to zero thus cancelling the requests.

Remarks:

none

CMP A INTMEM

OP-Code	Arg	Words	Cycles
4A*u8*	-	1	2

Function:

(Discarded) ← ACCU - INTMEM(u8);

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

Compare ACCU and INTMEM.

The content of INTMEM (addressed by the lower 8 bits of the OP-Code) is subtracted from the ACCU. The result is discarded (not written to the ACCU) but the flags are set like in a normal SUB instruction. The flags are affected as follows:

ZERO: set when the result is zero, otherwise cleared

NEG: set when d15 of result is one, otherwise cleared

OVER: set when result overflows into d15, otherwise cleared

CARRY: (=borrow) set when result is less then zero, otherwise cleared

The ZERO and the CARRY flag show the result of the comparison

ZERO	CARRY	Comparison result
1	x	A = INTMEM(u8)
0	x	A ≠ INTMEM(u8)
0	0	A > INTMEM(u8)
0	1	A < INTMEM(u8)

Remarks:

none

DEC INTMEM

OP-Code	Arg	Words	Cycles
46*u8*	-	1	2

Function:

INTMEM(u8) ← INTMEM(u8) - 1;

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

Decrement INTMEM.

The content of INTMEM (addressed by the lower 8 bits of the OP-Code) is decremented by one and stored back to the same location.

The flags are affected as follows:

ZERO: set when the result is zero, otherwise cleared

NEG: set when d15 of result is one, otherwise cleared

OVER: set to zero

CARRY: (=borrow) set when result is less then zero, otherwise cleared

Remarks:

This can be used for block transfers where the INTMEM serves as address pointer or as a loop counter.

INC INTMEM

OP-Code	Arg	Words	Cycles
47*u8*	-	1	2

Function:

INTMEM(u8) ← INTMEM(u8) + 1;

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

Increment INTMEM.

The content of INTMEM (addressed by the lower 8 bits of the OP-Code) is incremented by one and stored back to the same location.

The flags are affected as follows:

ZERO: set when the result is zero, otherwise cleared

NEG: set when d15 of result is one, otherwise cleared

OVER: set to zero

CARRY: set when result is bigger than 16 bits, otherwise cleared

Remarks:

This can be used for block transfers where the INTMEM serves as address pointer or as a loop counter.

JMPD U16

OP-Code	Arg	Words	Cycles
A000	U16	2	3

Function:

PC ← #U16

Affected Flags: none

Description:

Jump directly to address.

The jump address is the argument, i.e. the word following the opcode. The PC is loaded with the jump address and execution continues there.

Remarks:

none

JMPI INTMEM

OP-Code	Arg	Words	Cycles
A2*u8*	-	1	3

Function:

PC ← INTMEM(u8)

Affected Flags: none

Description:

Jump indirect via pointer in INTMEM.

The jump address is in INTMEM (which is addressed by the lower 8 bits of the OP-Code).

The jump address is loaded into the PC and execution continues there.

Remarks:

none

MOV INTMEM A

OP-Code	Arg	Words	Cycles
16*u8*	-	1	1

Function:

INTMEM(u8) ← A

PC ← PC + 1

Affected Flags: none

Description:

Move ACCU to INTMEM.

Copy ACCU to INTMEM (which is addressed by the lower 8 bits of the OP-Code)

Remarks:

none

MOV A INTMEM

OP-Code	Arg	Words	Cycles
50*u8*	-	1	2

Function:

A ← INTMEM(u8)

PC ← PC + 1

Affected Flags: none

Description:

Move INTMEM to ACCU.

Copy INTMEM (addressed by the lower 8 bits of the OP-Code) to ACCU.

Remarks:

none

MOV A STATUS

OP-Code	Arg	Words	Cycles
1400	-	1	1

Function:

A ← STATUS

PC ← PC + 1

Affected Flags: none

Description:

Move STATUS to ACCU.

The STATUS register reflects the stack pointer and the flags

d15 - d8	d7 - d5	d4	d3	d2	d1	d0
STACK POINTER	000	IE	ZERO	NEG	OVER	CARRY

Remarks:

none

MOV FLAGS A

OP-Code	Arg	Words	Cycles
1200	-	1	1

Function:

FLAGS ← A

PC ← PC + 1

Description:

Move ACCU to FLAGS.

Copy the 5 least significant bits of ACCU to the flag bits.

With this instruction all flags can be initialized to a specific value.

d15 - d5	d4	d3	d2	d1	d0
- - - - - - - - - - -	IE	ZERO	NEG	OVER	CARRY

Remarks:

Flags IE and CARRY can also be set and cleared with their dedicated instructions.

The stack pointer can only be loaded with its dedicated instruction (MOV SP A)

MOV IMASK A

OP-Code	Arg	Words	Cycles
0500	-	1	1

Function:

IMASK ← A

PC ← PC + 1

Affected Flags: none

Description:

Move ACCU to IMASK.

The number of bits in the IMASK register depends on the interrupt hardware but they always start at d0 and are arranged gapless. If there are e.g. four interrupt lines, the mask bits are d0 – d3 with d0 (IRQ0) having the highest and d3 (IRQ3) the lowest priority.

For an interrupt to be enabled its corresponding IMASK bit must be set to one.

Remarks:

The IMASK register is write only

MOV SP A

OP-Code	Arg	Words	Cycles
1300	-	1	1

Function:

SP ← A

PC ← PC + 1

Affected Flags: none

Description:

Move ACCU to SP.

Copy lower 8 bits of ACCU to stack pointer

Remarks:

To read back the content of register SP use the instructions MOV A STATUS and SWAP ACCU.

MOVD INTMEM U16

OP-Code	Arg	Words	Cycles
61*u8*	U16	2	2

Function:

INTMEM(u8) ← #U16

PC ← PC + 2

Affected Flags: none

Description:

Move direct value to INTMEM.

Load INTMEM (which is addressed by the lower 8 bits of the OP-Code) with 16-bit value.

Remarks:

none

MOVD A U16

OP-Code	Arg	Words	Cycles
6000	U16	2	2

Function:

A ← #U16

PC ← PC + 2

Affected Flags: none

Description:

Move direct value to ACCU.

Remarks:

none

MOVI INTMEM A

OP-Code	Arg	Words	Cycles
52u8	-	1	2

Function:

INTMEM ((low) INTMEM (u8)) ← A

PC ← PC + 1

Affected Flags: none

Description:

Move indirect ACCU to INTMEM.

The INTMEM argument is not the target address itself but a pointer to it. At first this pointer is fetched from INTMEM. Its lower 8 bits then address the actual INTMEM target location where the content of ACCU is copied to. The pointers upper 8 bits are ignored. This addressing mode is useful for moving and/or manipulating blocks of INTMEM in a loop by simply incrementing or decrementing the pointer.

Remarks:

none

MOVI A INTMEM

OP-Code	Arg	Words	Cycles
80*u8*	-	1	3

Function:

A ← INTMEM ((low) INTMEM (u8))

PC ← PC + 1

Affected Flags: none

Description:

Move indirect INTMEM to ACCU.

The INTMEM argument is not the source address itself but a pointer to it. At first this pointer is fetched from INTMEM. Its lower 8 bits then address the actual INTMEM source location whose content is copied to the ACCU. The pointers upper 8 bits are ignored. This addressing mode is useful for moving and/or manipulating blocks of INTMEM in a loop by simply incrementing or decrementing the pointer.

Remarks:

none

MOVXI INTMEM A

OP-Code	Arg	Words	Cycles
82*u8*	-	1	3

Function:

EXTMEM (INTMEM (u8)) ← A

PC ← PC + 1

Affected Flags: none

Description:

Move ACCU to EXTMEM addressed by pointer in INTMEM.

At first the INTMEM is addressed by the lower 8 bits of the OP-Code. This memory location serves as a pointer. It points to the actual EXTMEM target address where the content of ACCU is copied to. The indirect addressing can be used to move and/or manipulate blocks of EXTMEM in a loop by incrementing or decrementing the pointer.

Remarks:

none

MOVXI A INTMEM

OP-Code	Arg	Words	Cycles
83*u8*	-	1	3

Function:

A ← EXTMEM (INTMEM (u8))

PC ← PC + 1

Affected Flags: none

Description:

Move EXTMEM addressed by INTMEM to ACCU.

At first the INTMEM is addressed by the lower 8 bits of the OP-Code. This memory location serves as a pointer. It points to the actual EXTMEM location whose content is copied to ACCU. The indirect addressing can be used to move and/or manipulate blocks of EXTMEM in a loop by incrementing or decrementing the pointer.

Remarks:

none

NOP

OP-Code	Arg	Words	Cycles
0000	-	1	1

Function:

PC ← PC + 1

Affected Flags: none

Description:

No operation.

Only the PC is incremented by one. This instruction can be used for short delays or to fill unused memory.

Remarks:

none

NOT A

OP-Code	Arg	Words	Cycles
2F00	-	1	1

Function:

ACCU ← **NOT** ACCU

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER

Description:

Negate ACCU.

All bits of ACCU are negated and the result is stored into ACCU.

The flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set to zero

Remarks:

Two consecutive NOT A leave the ACCU unchanged but set / clear the flags ZERO and NEG. This can be used to test if the ACCU contains zero or a negative value.

OR A INTMEM

OP-Code	Arg	Words	Cycles
4D*u8*	-	1	2

Function:

ACCU ← ACCU **OR** INTMEM(u8)

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER

Description:

Or ACCU with INTMEM.

The contents of ACCU and INTMEM (addressed by the lower 8 bits of the OP-Code) are ORed bit-wise and the result is stored into the ACCU.

The flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set to zero

Remarks:

none

POP A

OP-Code	Arg	Words	Cycles
5100	-	1	2

Function:

SP ← SP + 1

ACCU ← INTMEM(SP)

PC ← PC + 1

Affected Flags: none

Description:

Pop ACCU.

At first the stack pointer is incremented by one. Next the content of INTMEM (addressed by the stack pointer) is copied into ACCU.

Remarks:

none

PUSH A

OP-Code	Arg	Words	Cycles
1500	-	1	1

Function:

INTMEM(SP) ← ACCU

SP ← SP - 1

PC ← PC + 1

Affected Flags: none

Description:

Push ACCU.

At first the content of ACCU is copied into INTMEM (addressed by the stack pointer). Next the stack pointer is decremented by one.

Remarks:

none

RETI

OP-Code	Arg	Words	Cycles
8500	-	1	3

Function:

SP ← SP + 1

IIPx ← 0

PC ← INTMEM(SP)

Affected Flags: none

Description:

Return from interrupt.

At first the stack pointer is incremented by one and the execution of the current interrupt is ended by clearing the correspondent IIPx-Flag (Interrupt In Progress). Next the content of INTMEM (addressed by the stack pointer) is copied into PC and execution continues from there.

Remarks:

A valid return address must be on the stack from an interrupt call

RETS

OP-Code	Arg	Words	Cycles
8400	-	1	3

Function:

SP ← SP + 1

PC ← INTMEM(SP)

Affected Flags: none

Description:

Return from subroutine.

At first the stack pointer is incremented by one. Next the content of INTMEM (addressed by the stack pointer) is copied into PC and execution continues from there.

Remarks:

A valid return address must be on the stack from a previous CALL instruction

ROL A

OP-Code	Arg	Words	Cycles
2800	-	1	1

Function:

CARRY ← ACCU(15 ← 14 ← … ← 2 ← 1 ← 0) ← CARRY

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

Rotate Left Accumulator through CARRY.

The ACCU is shiftet left one bit position. Its lowest bit receives the content of the CARRY. Its highest bit in turn goes into the CARRY.

The other flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set to zero

Remarks:

The CARRY can be set or cleared beforehand to insert a one or a zero.

ROR A

OP-Code	Arg	Words	Cycles
2900	-	1	1

Function:

CARRY → ACCU(15 → 14 → … → 2 → 1 → 0) → CARRY

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

Rotate Right Accumulator through CARRY.

The ACCU is shiftet right one bit position. Its highest bit receives the content of the CARRY. Its lowest bit in turn goes into the CARRY.

The other flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set to zero

Remarks:

The CARRY can be set or cleared beforehand.

SET C

OP-Code	Arg	Words	Cycles
1100	-	1	1

Function:

CARRY ← 1

PC ← PC + 1

Affected Flags: CARRY

Description:

Set CARRY flag.

This can be used to prepare shift and rotate instructions.

Remarks:

none

SET IE

OP-Code	Arg	Words	Cycles
0300	-	1	1

Function:

IE ← 1

PC ← PC + 1

Affected Flags: IE

Description:

Set Interrupt Enable flag.

All pending and enabled interrupts will be accepted in the order of their priority.

(INT0 = highest priority)

Remarks:

none

SHL A

OP-Code	Arg	Words	Cycles
2A00	-	1	1

Function:

CARRY ← ACCU(15 ← 14 ← … ← 2 ← 1 ← 0) ← 0

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

Shift Left Accumulator into CARRY.

The ACCU is shiftet left one bit position. Its lowest bit receives a zero. Its highest bit in turn goes into the CARRY.

The other flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set to zero

Remarks:

SHL A is equivalent to multiply by 2

SHR A

OP-Code	Arg	Words	Cycles
2B00	-	1	1

Function:

0 → ACCU(15 → 14 → … → 2 → 1 → 0) → CARRY

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

Shift Right Accumulator into CARRY.

The ACCU is shiftet right one bit position. Its highest bit receives a zero. Its lowest bit in turn goes into the CARRY.

The other flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set to zero

Remarks:

SHR A is equivalent to division by 2

SUB A INTMEM

OP-Code	Arg	Words	Cycles
42*u8*	-	1	2

Function:

ACCU ← ACCU - INTMEM(u8);

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

The content of INTMEM (addressed by the lower 8 bits of the OP-Code) is subtracted from ACCU and the result is stored into ACCU.

The flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set when sign overflows, otherwise cleared
- CARRY (=borrow) set when result is less then zero, otherwise cleared

Remarks:

This instruction does **not** subtract the current CARRY. It can be used for a 16-bit subtraction or to subtract the lowest two words in a higher precision (e.g. 64-bit) subtraction.

SUBB A INTMEM

OP-Code	Arg	Words	Cycles
40*u8*	-	1	2

Function:

ACCU ← ACCU - INTMEM(u8) - CARRY;

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER, CARRY

Description:

Subtract with borrow.

The content of INTMEM (addressed by the lower 8 bits of the OP-Code) and CARRY are subtracted from ACCU and the result is stored into ACCU.

The flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set when sign overflows, otherwise cleared
- CARRY (=borrow) set when result is less then zero, otherwise cleared

Remarks:

SUBB is normally used in higher precision (e.g. 64-bit) subtractions. The lowest two words are subtracted with the SUB instruction. Then the higher words are subtracted with the SUBB instruction, taking the carries (borrows) into account.

SWAP A

OP-Code	Arg	Words	Cycles
2500	-	1	1

Function:

ACCU(15 .. 8) ← ACCU(7 .. 0), ACCU(7 .. 0) ← ACCU(15 .. 8)

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER

Description:

SWAP ACCU

Swap the high and low bytes of ACCU.

The flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set to zero

Remarks:

XOR A INTMEM

OP-Code	Arg	Words	Cycles
4E*u8*	-	1	2

Function:

ACCU ← ACCU **XOR** INTMEM(u8)

PC ← PC + 1

Affected Flags: ZERO, NEG, OVER

Description:

The contents of ACCU and INTMEM (addressed by the lower 8 bits of the OP-Code) are XORed bit-wise and the result is stored into the ACCU.

The flags are affected as follows:

- ZERO set when the result is zero, otherwise cleared
- NEG set when d15 of result is one, otherwise cleared
- OVER set to zero

Remarks:

none