ANC216

A 16-bit architecture

Simone Ancona

Introduction

The ANC216 architecture is a 16-bit architecture created for educational purposes. It can be useful for studying and understanding how computers work.

Definitions

This section contains all the definitions and abbreviations that can be found in the article.

- BP: Base Pointer.
- Bus: a computer coMTU nication system used to connect components and peripherals.
- Byte: 8 bits.
- CPU: Central Processing Unit, is the main processor in a computer.
- EINR: External Interrupt.
- EMEM: External Memory.
- GPR: General Purpose Register.
- HEX: Hexadecimal.
- IMEM: Internal Memory.
- INR: Interrupt.
- ISA: Instruction Set Architecture.
- IO: Input/Output.
- MTU: Memory Table Unit.
- NMI: non-maskable interrupt.
- OPC: Operation Code.
- PC: Program Counter.
- RAM: Random Access Memory.
- ROM: Read Only Memory.
- SP: Stack Pointer.
- Word: 16 bits.

Architecture

In computer science an architecture is the structure and the organization of a CPU and its internal components. The ANC216 architecture was designed for educational purposes but it can be used for small embedded systems.

An ANC216 microprocessor consists of several components, such as memories, buses and others.

Registers

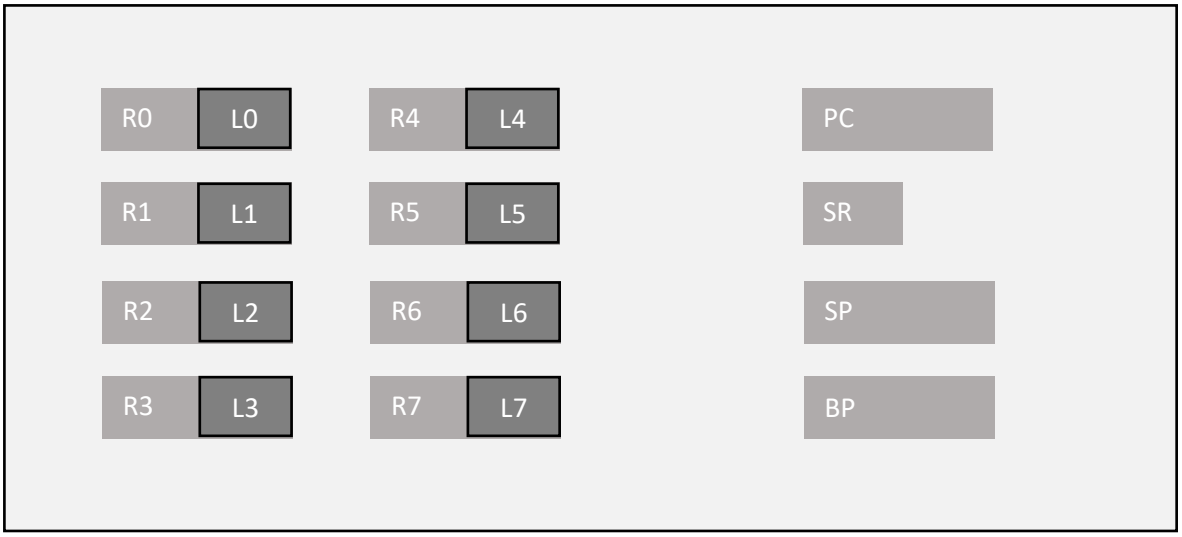
Registers are tiny memories used to temporary store data such as addresses, numbers or characters to print on the screen.

There are different categories of registers. We call general purposes registers, or GPRs, those registers that are not used for a specific use but used to store data and to compute arithmetic, logic and other operations. Then we have registers used for specific purposes.

An ANC216 microprocessor has 8 GPRs named R0 to R7. Each of these registers is 16-bit long and has a low part of 8-bit that is named Lx where x is the number of the register.

There are 4 other registers used for specific purposes:

- The PC, Program Counter, is a 16-bit register that holds the address of the next instruction to be executed.
- The SR, Status Register, is an 8-bit register used to store additional information about the result of the current instruction and the status of the CPU. This register consists of a set of individual bits each representing a specific condition, the flags shown here are in ordered like in the SR from bit 7 to bit 0:
  - **N**egative: set if the result is negative.
  - **O**verflow: set if the result of an integer operation is out of the range.
  - **I**nterrupts: set to enable interrupts, (NMI, syscall and reset interrupts are always enabled).
  - **T**imer interrupt: set to enable watch interrupt (go to interrupts section for more).
  - **S**ystem privileges: set to enable system privileges. This flag can be modified only if is set.
  - Reserved, always 1.
  - **Z**ero: set if the result is zero.
  - **C**arry: set if the result has a carry.
- The SP, Stack Pointer, is a 16-bit register used to refer to the top of the stack.
- The BP, Base Pointer, is a 16-bit register used to refer to the base of the stack.



The ANC216 has also a hidden 16-bit register used by the internal timer.

Buses

A bus is a connection between internal CPU's components or peripherals. In an ANC216 microprocessor we can find 3 types of buses.

- Address bus: 16-bit bus used to specify an address of a memory cell or an IO device, a memory cell stores 8-bit, a byte.
- Data bus: 16-bit used to transfer data among microprocessor's components or peripherals.
- Control bus: an 8-bits wide bus.
  - The wire 0 is used to specify an internal (0) or an external (1) connection.
  - The wire 1 is used to specify read (0) or write (1).
  - The wire 2 is the request to the bus arbiter (used only when the CPU requires an external connection).
  - The wire 3 is the bit length of the data 8-bit (0) or 16-bit (1).
  - The wire 4 is the information control bit (used only when the CPU requires an external connection).
  - The wire 5 is the high privileges request to the bus arbitration unit (used only in external connections).
  - The wire 6 tells the CPU if the bus is used by an external device.
  - Finally, the wire 7 is an additional information wire for external devices.

  An external connection does not necessarily provide data the next clock cycle because the bus could be busy.

  When an external device responds to a previous data request of the CPU, wire 1 is set to 0, wire 3 is set according to data length, wire 4 is set to 0.

  When an external device responds to a previous information request of the CPU, wire 1 is set to 0, wire 4 is set to 1.

  Finally, when an external device wants to know the CPU ID, wire 1 is set to 1, wire 4 is set to 1.

Instructions

An instruction is a command that directs the CPU to perform a certain operation. There are different kind of operations that the CPU can perform:

- Arithmetic operations, such as the sum, the subtraction, increment, decrement.
- Logical operations such as and, or, xor, not.
- Shift operations.
- Transfer operations used to transfer some data from a register to another.
- Load operations used to load data from memory.
- Store operations used to store data in memory.
- Stack instructions, used to perform stack operations.
- Flag instructions that can modify the SR.
- Comparison instructions, used to compare data.
- Jumps, used to divert the flow of program execution.
- Software interrupts.
- IO instructions.

In ANC16 an instruction is a word (16-bit long) and consists of the addressing mode (first byte big endian) that specify how the data is fetched (go to addressing modes for more) and the opcode used to specify the instruction.

Memory

An ANC216 microprocessor incorporates a 64KB RAM, called IMEM, used to store the current program in execution, the operating system, data or information regarding IO devices. It also contains a 256 bytes ROM used to store the firmware. The firmware is the software that loads the operating system into the RAM from the EMEM (External Memory). The OS is loaded into memory `0x0000` to `0x3000`.

The memory is mapped as follows:

- From `0x0000` to `0x0001` there is the OS entry point vector where is stored the pointer to the first routine of the OS.
- From `0x0002` to `0x0003` there is the external interrupts vector where is stored the pointer to the interrupt handler.
- From `0x0004` to `0x0005` there is the NMI vector, where is stored the address to the NMI handler.
- From `0x0006` to `0x0007` there is the system call vector, where is stored the address to the system call handler.
- From `0x0008` to `0x0009` there is the timer interrupt vector, where is stored the address to the timer interrupt handler.
- From `0x000A` to `0x000B` there is the shutdown interrupt vector, where is stored the address to the shutdown handler.

- In `0x000C` there is the system stack pointer, when an interrupt is issued, the SP is set to the value stored in this cell.
- From `0x000D` to `0x2FFF` (arbitrary) there is the memory reserved to the operating system.
- From `0x3000` to `0x31FF` there is the system stack.
- From `0x3200` (arbitrary) to `0xFEFF` free memory.
- From `0xFF00` to `0xFFFF` the ROM is mapped. The ROM contains the firmware which is a program, in this case used to boot the PC.

The ANC216 architecture has a unit called MTU (Memory Table Unit) which allows the OS to map user programs to avoid memory access conflicts.

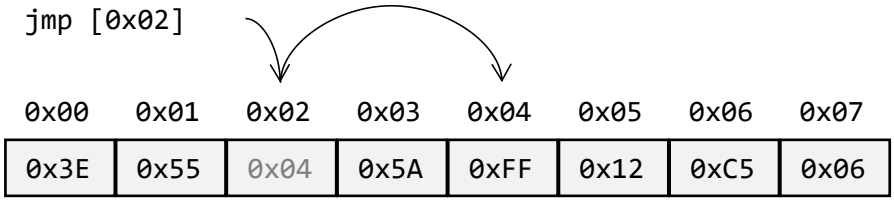
The MTU is like an array of words:

IMEM lower index	IMEM higher index	EMEM lower index	EMEM higher index	Stack base	Stack top
16-bit	16-bit	16-bit	16-bit	16-bit	16-bit

Addressing modes

An addressing mode specify how operands of an instruction are accessed or fetched from the memory. There are different categories of addressing modes in ANC216:

- Implied: the instruction does not take any operand.
- Immediate: the operand is a constant value.
- Register access: the operand value is stored in one register.
- Register-to-register: the operands are two registers.
- Memory related addressing modes: are those addressing modes that describe how to access memory:
  - Absolute addressing: when the argument of the instruction is the address of a cell in memory. The absolute addressing is mapped according to the MTU .
  - Absolute indexed: from the absolute address, a value stored in a register is added.
  - Indirect addressing: when the argument of the instruction is an address stored in a cell in memory referred by an absolute address. Example:



  - Indirect indexed: from the indirect, a value stored in a register is added to the final address.
  - Relative to PC: when the argument of the instruction is the address of a cell calculated by adding an 8-bit signed value (constant or register) to the PC.
  - Relative to SP: similar to the previous addressing mode, the address i calculated by adding an 8-bit singed value (constant or register) to the SP.

- Register-to-memory: one operand is a register and the other one is a memory related addressing:
  - Register-absolute: one operand is a register and the other is an absolute address. The direction is specified by the instruction. Example:
    `load r6, & 0x30FF` This instruction loads in the register R6 the value stored in `0x30FF`.
    `store & 0x30FF, r3` This instruction is the opposite one, the value on R3 is stored in `0x30FF`.
  - Register-immediate: one operand is a register and the other is a constant value. Example:
    `load r0, 10` This instruction loads the number 10 (`0x0A`) in R0.
  - Register-relative to PC.
  - Register-relative to SP.

Note that in register-to-memory the register can be a low part of the register but in register-to-register it can be only a full-size register.

As mentioned above, an instruction consists of the addressing mode byte and the opcode byte. The addressing mode byte follows the pattern below:

Byte	Name	Assembly example	Argument size (bytes)	Note
Implied				
`00 000 000`	Implied	`ret`	0	
Immediate				
`00 001 000`	Immediate	`push byte 24`	1	
`00 010 000`	Immediate	`push word 24`	2	
Register access				
`00 xxx 001`	Register access	`trt r5`	0	xxx is the register id
`00 xxx 010`	Low register access	`ldsr l5`	0	xxx is the register id
Register to register				
`01 xxx yyy`	Register to register	`tran r4, r0`	0	xxx is destination (r4) and yyy is source (r0)
Memory related				
`10 000 000`	Absolute	`jmp & 0x0123`	2	
`10 xxx 001`	Absolute indexed	`jmp & 0x0123 + l2`	2	xxx is the low part register id
`10 000 100`	Relative to PC	`jmp * 23`	1	
`10 001 100`	Relative to BP	`jmp & bp + 10`	1	
`10 xxx 101`	Relative to PC with register	`jmp * l0`	0	xxx is the low part register id
`10 xxx 110`	Relative to BP with register	`jmp & bp + l4`	0	xxx is the low part register id

Indirect				
10 000 010	Indirect	jmp [0x0123]	2	
10 xxx 011	Indirect indexed	jmp [0x0123] + l0	2	xxx is the low part register id
Immediate to memory				
00 000 011	Immediate to Relative BP	store & bp + 5, 20	2 (1 + 1)	(1 + 1) means: 1 for the index value and 1 for the argument
00 001 011	Immediate to Relative BP	store & bp - 3, 256	3 (1 + 2)	(1 + 2) means: 1 for the index value and 2 for the argument
00 xxx 100	Immediate to Relative BP with register	store & bp + l1, 7	1	xxx is the low part register id
00 xxx 101	Immediate to Relative BP with register	store & bp - l4, 44	2	xxx is the low part register id
00 000 110	Immediate to absolute	store & 0x0456, 20	3 (2 + 1)	(2 + 1) means: 2 for the address value and 1 for the argument
00 001 110	Immediate to absolute	store & 0xFF00, 450	4 (2 + 2)	(2 + 2) means: 2 for the address value and 2 for the argument
00 xxx 111	Immediate to absolute indexed	store & 0xF + l2, 5	3 (2 + 1)	xxx is the low part register id
Register to memory				
11 xxx 000	Register-absolute	load r4, & 0x0123	2	xxx is the register id
11 xxx 001	Register-immediate	load r0, 56	2	xxx is the register id
11 xxx 010	Register-PC relative	load r3, * 10	1	xxx is the register id
11 xxx 011	Register-BP relative	load r7, & bp - 13	1	xxx is the register id
11 xxx 100	Low register-absolute	store & 0x0123, l0	2	xxx is the register id
11 xxx 101	Low register-immediate	load l0, byte 32	1	xxx is the register id
11 xxx 110	Low register-PC relative	load l3, * -5	1	xxx is the register id
11 xxx 111	Low register-BP	load l7, & bp + 5	1	xxx is the register id

relative

Interrupts

An interrupt is an internal or external signal that interrupt the execution flow of the CPU. An interrupt is handled by a specific routine of the operating system called interrupt handler.

There are two categories of interrupts: hardware and software.

Hadware interrupts:

- EINR: External interrupt. The interrupt is handled by the routine with the address stored in `0x0002`. The standard interrupt procedure is implemented. R0, R1 and L2 are pushed. The external device mapped address is stored in R0, data are stored in R1 and the request type is stored in L2. The request type can be:
  - `0x00`: an external device requires the CPU ID info.
  - `0x01`: an external device responded to a previous CPU request sending data.
  - `0x02`: an external device responded to a previous CPU request sending data and the additional wire in the bus control is set.
  - `0x03`: an external device responded to a previous CPU information request.
- NMI: Non-maskable interrupts. The interrupt is handled by the routine with the address stored in `0x0004`. Like any other interrupt, the standard procedure is implemented. L0 is pushed. The NMI code is stored in R0. This is the list of NMI codes:
  - `0x00`: for unrecognized or version-incompatible opcodes.
  - `0x01`: when a user application* uses a high privileges instruction (the system privileges flag indicates whether the privileges are granted or not).
  - `0x02`: when a user application tries to access an area of IMEM that does not belong to it.
  - `0x03`: when a user application tries to access an area of EMEM that does not belong to it.
  - `0x04`: when a user application has a stack overflow (the SP is grater than the Stack Top in MTU).
  - `0x05`: when the soft reset pin is triggered.
- Hard reset: when the hard reset pin is triggered. The standard interrupt procedure is not implemented, the CPU runs the firmware to reboot.
- Timer interrupt: when the timer expires, the interrupt handler address is stored in `0x0008`. The standard procedure is implemented.
- Shutdown interrupt: when the shutdown pin is triggered. The standard procedure is not implemented. The address to the routine in stored in `0x000A`.

Software interrupts:

- System call: this interrupt is issued by the `syscall` instruction. The standard procedure is implemented.

The standard interrupt procedure is a procedure used to save the current CPU state to then load it once the interrupt handling is finished. The procedure is the following: The BP is set to `0x3000`, the SP stored in `0x31FF` and then is set to the value stored in `0x000C`, PC, SR and other registers are

pushed (which registers are pushed depends on the interrupt). This procedure is automatic, the restore procedure must be done manually.

*user application, when the System Privileges flag is clear.

Instruction Set Architecture

The instruction set architecture, or ISA, is the set of instructions recognized by the microprocessor. We show the summary table of the instructions below:

Instruction marked with * require system privileges.

OPC	Mnemonic	Description	Type
0x00	kill*	Stop the execution flow of the CPU	Interrupt
0x01	reset*	Issue a soft reset NMI	Interrupt
0x02	cpuid	Get CPU information	Other
0x03	syscall	Make a system call	Interrupt
0x04	call	Call a routine	Stack
0x05	ret	Return from a routine	Stack
0x06	push	Push onto the stack	Stack
0x07	pop	Pop from the stack	Stack
0x08	phpc	Push the PC	Stack
0x09	popc	Pop the PC	Stack
0x0A	phsr	Push SR	Stack
0x0B	posr	Pop SR	Stack
0x0C	phsp	Push SP	Stack
0x0D	posp	Pop SP	Stack
0x0E	phbp	Push BP	Stack
0x0F	pobp	Pop BP	Stack
0x10	seti*	Set I flag	Flag
0x11	sett*	Set T flag	Flag
0x12	sets*	Set S flag	Flag
0x13	clri*	Clear I flag	Flag
0x14	clrt*	Clear T flag	Flag
0x15	clrs*	Clear S flag	Flag
0x16	clrn	Clear N flag	Flag
0x17	clro	Clear O flag	Flag
0x18	clrc	Clear C flag	Flag
0x19	ireq	Make an information EMEM device request	Asynchronous IO
0x1A	req	Make an EMEM read request	Asynchronous IO

0x1B	write	Make an EMEM write request	Asynchronous IO
0x1C	hreq*	Make a high privileges EMEM read request	Asynchronous IO
0x1D	hwrite*	Make a high privileges EMEM write request	Asynchronous IO
0x1E	read	Make a synchronous EMEM read request	IO
0x1F	pareq	Used to trigger the additional info wire in control bus	IO
0x20	cmp	Compare	Compare
0x21	careq	Used to clear the additional info wire in control bus	IO
0x22	jmp	Jump	Jump
0x23	jeq, jz	Jump if equal, jump if zero	Jump
0x24	jne, jnz	Jump if not equal, jump if not zero	Jump
0x25	jge	Jump if grater or equal	Jump
0x26	jgr	Jump if grater	Jump
0x27	jle	Jump if less or equal	Jump
0x28	jls	Jump if less	Jump
0x29	jo	Jump if overflow	Jump
0x2A	jno	Jump if not overflow	Jump
0x2B	jn	Jump if negative	Jump
0x2C	jnn	Jump if not negative	Jump
0x2D	inc	Increment	Arithmetic
0x2E	dec	Decrement	Arithmetic
0x2F	add	Addition	Arithmetic
0x30	sub	Subtraction	Arithmetic
0x31	neg	Two’s complement	Arithmetic
0x32	and	And logical operation	Logical
0x33	or	Or logical operation	Logical
0x34	xor	Xor logical operation	Logical
0x35	not	One’s complement	Logical
0x36	sign	Set N if the sign bit is set	Flag
0x37	shl	Shift left	Shift
0x38	shr	Shift right	Shift
0x39	par	Set Z if the number of 1s is even	Flag
0x3A	load	Load from memory	Load
0x3B	store	Store in memory	Store
0x3C	tran	Transfer data between registers	Transfer
0x3D	swap	Swap data between registers	Transfer

0x3E	ldsr	Load SR	Load
0x3F	ldsp	Load SP	Load
0x40	ldbp	Load BP	Load
0x41	stsr	Store SR	Store
0x42	stsp	Store SP	Store
0x43	stbp	Store BP	Store
0x44	trsr	Transfer SR to a register	Transfer
0x45	trsp	Transfer SP to a register	Transfer
0x46	trbp	Transfer BP to a register	Transfer
0x50	sili*	Set IMEM lower index in MTU	MTU
0x51	sihi*	Set IMEM higher index in MTU	MTU
0x52	seli*	Set EMEM lower index in MTU	MTU
0x53	sehi*	Set EMEM higher index in MTU	MTU
0x54	sbp*	Set stack base in MTU	MTU
0x55	stp*	Set stack top in MTU	MTU
0x56	tili	Transfer IMEM lower index into a register	MTU
0x57	tihi	Transfer IMEM higher index into a register	MTU
0x58	teli	Transfer EMEM lower index into a register	MTU
0x59	tehi	Transfer EMEM higher index into a register	MTU
0x5A	tbp	Transfer stack base into a register	MTU
0x5B	ttp	Transfer stack top into a register	MTU
0x60	time*	Set the internal timer	Timer
0x61	tstart*	Run the timer	Timer
0x62	tstop*	Stop the timer	Timer
0x63	trt	Transfer timer into register	Transfer

The instruction is formed by concatenating the addressing mode byte and the OPC. For example: `write & 0xFF00, r4` This instruction makes an external IO device request for writing data stored in R4 to the external device mapped in `0xFF00`.

The binary will be the concatenation between its addressing and OPC (big endian):

The addressing is a register-memory, more specifically register-absolute (even though we use the external memory), the register is the R4, its ID is `0b100` (4 in binary) so the addressing will be: `11 100 000` (go to addressing modes for more).

The OPC is `0x17` (`0001 0111` in binary).

So, the instruction `write & 0xFF00, r4` will be `1110 0000 0001 0111 1111 1111 0000 0000` (`E017 FF00` in hex) where:

- The part in black is the addressing mode.

- The part in dark gray is the OPC (operation code).
- And the part in light gray is the operand (`0xFF00`).

System calls

A system call is a software interrupt managed by the operating system and is used by programs to read and modify resources that only the operating system can access such as video memory. The system call inr is issued using the `syscall` instruction. Arguments are passed via registers.

Below is the table of standard system calls:

Name	L0	R1	R2	L2	R3	L3	R7
`exit`	`0x00`	Exit code					
`fopen`	`0x01`	File path (string)	File path string size (8-bit integer)		File descriptor id (pointer to 8-bit cell)		Return code
`fclose`	`0x02`	File descriptor id (8-bit integer)					Return code
`fread`	`0x03`	File descriptor id (8-bit integer)	Buffer size (8-bit integer)		Buffer (pointer to 8-bit cells buffer)		Return code
`fwrite`	`0x04`	File descriptor id (8-bit integer)	Write mode: write (`0x80`) / read (`0x00`)	Buffer size (8-bit integer)	Buffer (pointer to 8-bit cells buffer)		Return code
`print`	`0x05`	Message (string)	Message size (8-bit integer)		Stream: stdout (`0x00`) / stderr (`0x01`)		Return code
`getl`	`0x06`	Buffer size (8-bit integer)	Buffer (pointer to 8-bit cells)				String size
`sleep`	`0x07`	Milliseconds (16-bit integer)					
`listenkey`	`0x08`	Event handler (pointer to a routine)					Key code
`reqh`	`0x09`						Return code
`malloc`	`0x0A`	Size (8-bit integer)					Address to memory
`dealloc`	`0x0B`	Address to memory (pointer)					Return code
`frm`	`0x0C`	File path (string)	File path string size (8-bit integer)				Return code
`dirrm`	`0x0D`	Dir path (string)	Dir path string size				Return code

			(8-bit integer)			
`mkdir`	`0x0E`	Dir path (string)	New dir name (string)	Dir path string size (8-bit)	New dir name string size (8-bit)	Return code

This table shows the standard syscalls and which registers are needed to pass arguments. An OS could have more syscalls or different one.

Below is the description for each syscall:

- `exit`: The exit system call is used to kill the execution of the program.
- `fopen`: Is used to open files or streams (the file-name can be `stdout` or `stderr` for opening streams).
- `fclose`: Is used to close and save files.
- `fread`: Is used to read from streams.
- `fwrite`: Is used to write into streams.
- `print`: Is used to print strings in the standard output or standard error stream.
- `getl`: Is used to get lines from the standard input stream. The return value is the length of the string read.
- `sleep`: The sleep system call is used to pause the execution of the program.
- `listenkey`: Is used to handle keyboard events.
- `reqh`: Is used to request high privileges (or system privileges).
- `malloc`: Is used to allocate memory dynamically. Return the address to the allocated memory or `0x0000` in case of error.
- `dealloc`: Is used to free memory dynamically.
- `frm`: Is used to remove files from the file system.
- `dirrm`: Is used to remove directories from the file system.
- `mkdir`: Is used to create directories in the file system.

The return code is `0x00` for no errors, otherwise another value.

Hard reset interrupt

When the CPU is reset or turned on, the PC is set to `0xFF00`, where the firmware resides. The SR is set to: `n o I T S 1 z c` (upper-case means set). The firmware loads the OS from the EMEM starting at the address `0x0000` to the address `0x2FFF` (included), all other addresses in the EMEM are arbitrary and may depend on the operating system.

Instructions in detail

Below are all the instructions in detail with compatible addressing modes and examples in assembly.

Add

Description	Supported addressing modes	This operation may change the flags	Operations
Add a value to a register	• Register to register • Register to memory	• N • O • Z • C	• Ra = Ra + Rb • R = R + M

And

Description	Supported addressing modes	This operation may change the flags	Operations
Bit-wise and between a register and a operand	• Register to register • Register to memory	• N • Z	• Ra = Ra & Rb • R = R & M

Call

Description	Supported addressing modes	This operation may change the flags	Operations
Call a routine	• Memory related		• Push PC + offset; Push SR, JMP to routine BP = SP

Careq

Description	Supported addressing modes	This operation may change the flags	Operations
Clear additional information request (clear the Z flag)	• Implied	• Z	

Clrc

Description	Supported addressing modes	This operation may change the flags	Operations
Clear carry flag	• Implied	• C	• C = 0

Clri

Requires system privileges.

Description	Supported addressing modes	This operation may change the flags	Operations
Clear interrupt flag	• Implied	• I	• I = 0

Clrn

Description	Supported addressing modes	This operation may change the flags	Operations
Clear negative flag	• Implied	• N	• N = 0

Clro

Description	Supported addressing modes	This operation may change the flags	Operations
Clear overflow flag	• Implied	• O	• O = 0

Clrs

Requires system privileges.

Description	Supported addressing modes	This operation may change the flags	Operations
Clear system privileges flag	• Implied	• S	• S = 0

Clrt

Requires system privileges.

Description	Supported addressing modes	This operation may change the flags	Operations
Clear timer interrupt flag	• Implied	• T	• T = 0

Cmp

Operation	Carry	Zero	Negative	Overflow
Register > operand	0	0	Sing bit of result R - O	Overflow of R - O
Register == operand	0	1	0	Overflow of R - O

Register < operand	1	0	Sign bit of result R - O	Overflow of R - O

Description	Supported addressing modes	This operation may change the flags	Operations
Compare two operands.	• Register to register • Register to memory	• N • O • Z • C	• R == M • Ra == Rb

Cpuid

Description	Supported addressing modes	This operation may change the flags	Operations
Get information about the CPU	• Implied		• R0 = `0x8000` (the ANC216 id, may vary from version to version)

Dec

Description	Supported addressing modes	This operation may change the flags	Operations
Decrement register	• Register access	• N • O • Z • C	• R = R - 1 • L = L - 1

Hreq

Requires system privileges.

Description	Supported addressing modes	This operation may change the flags	Operations
Make a high privileges IO read request.	• Memory related • Register access (get the address from registers)		

Hwrite

Requires system privileges.

Description	Supported addressing modes	This operation may change the flags	Operations
Make a high privileges IO write request	• Register to memory • Immediate to memory		

Inc

Description	Supported addressing modes	This operation may change the flags	Operations
Increment register	• Register access	• N • O • Z • C	• R = R + 1 • L = L + 1

Ireq

Description	Supported addressing modes	This operation may change the flags	Operations
Make an information IO device request	• Memory related • Register access (store the address)		After the response: • R0 = Device ID • R1 = Device address

Jeq, Jz

Description	Supported addressing modes	This operation may change the flags	Operations
Jump if Z is set (jump if equal)	• Memory related • Indirect		• If (Z) jump

Jge

Description	Supported addressing modes	This operation may change the flags	Operations
Jump if N == O (jump if grater or equal)	• Memory related • Indirect		• If (N == O) jump

Jgr

Description	Supported addressing modes	This operation may change the flags	Operations
Jump if N == O and Z == 0 (jump if grater)	• Memory related • Indirect		• If (N == O and Z == 0) jump

Jle

Description	Supported addressing modes	This operation may change the flags	Operations
Jump if N != O and Z == 1 (jump if less or equal)	• Memory related • Indirect		• If (N != O and Z == 1) jump

Jls

Description	Supported addressing modes	This operation may change the flags	Operations
Jump if N != O (jump if less)	• Memory related • Indirect		• If (N != O) jump

Jmp

Description	Supported addressing modes	This operation may change the flags	Operations
Jump	• Memory related • Indirect		• Jump

Jn

Description	Supported addressing modes	This operation may change the flags	Operations
Jump if N is set	• Memory related • Indirect		• If (N) jump

Jne, Jnz

Description	Supported addressing modes	This operation may change the flags	Operations
Jump if Z is clear (jump	• Memory related		• If (!Z) jump

if not equal)	• Indirect		

Jnn

Description	Supported addressing modes	This operation may change the flags	Operations
Jump if N is clear	• Memory related • Indirect		• If (!N) jump

Jno

Description	Supported addressing modes	This operation may change the flags	Operations
Jump if O is clear	• Memory related • Indirect		• If (!O) jump

Jo

Description	Supported addressing modes	This operation may change the flags	Operations
Jump if O is set	• Memory related • Indirect		• If (O) jump

Kill

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Kill the execution of the CPU	• Implied		

Ldbp

Description	Supported addressing modes	This operation may change the flags	Operations
Load BP	• Register access • Memory related	• N • Z	• BP = R • BP = M

Ldsp

Description	Supported addressing modes	This operation may change the flags	Operations
Load SP	• Register access • Memory related	• N • Z	• SP = R • SP = M

Ldsr

Description	Supported addressing modes	This operation may change the flags	Operations
Load SR	• Register access (only 8-bit low) • Memory related	• N • Z	• SR = L • SR = M

Load

Description	Supported addressing modes	This operation may change the flags	Operations
Load a value from memory into register	• Register to memory	• N • Z	• R = M

Neg

Description	Supported addressing modes	This operation may change the flags	Operations
Two's complement	• Register access	• N • Z	• R = -R • L = -L

Not

Description	Supported addressing modes	This operation may change the flags	Operations
One's complement	• Register access	• N • Z	• R = ~R • L = ~L

Or

Description	Supported addressing modes	This operation may change the flags	Operations

Bit-wise or between a register and a operand	• Register to register • Register to memory	• N • Z	• Ra = Ra | Rb • R = R | M

Par

Description	Supported addressing modes	This operation may change the flags	Operations
Set Z if the number of 1s is even	• Register access • Memory related	• Z	• Parity R • Parity M

Pareq

Description	Supported addressing modes	This operation may change the flags	Operations
Prepare additional information request (set Z flag)	• Implied	• Z	

Phbp

Description	Supported addressing modes	This operation may change the flags	Operations
Push the BP	• Implied		• Push BP; SP += 2

Phpc

Description	Supported addressing modes	This operation may change the flags	Operations
Push the PC	• Implied		• Push PC; SP += 2

Phsp

Description	Supported addressing modes	This operation may change the flags	Operations
Push the SP	• Implied		• Push SP; SP += 2

Phsr

Description	Supported addressing modes	This operation may change the flags	Operations

Push the SR	• Implied		• Push SR; SP += 1

Pobp

Description	Supported addressing modes	This operation may change the flags	Operations
If the SP is grater than the BP, pop BP	• Implied	• N • Z	• SP -= 2; BP = & SP

Pop

Description	Supported addressing modes	This operation may change the flags	Operations
If the SP is grater than the BP, pop from the stack	• Register access	• N • Z	• SP -= 2; R = & SP • SP -= 1; L = & SP

Popc

Description	Supported addressing modes	This operation may change the flags	Operations
Pop PC (if SP > BP)	• Implied	• N • Z	• SP -= 2; PC = & SP

Posp

Description	Supported addressing modes	This operation may change the flags	Operations
Pop SP (if SP > BP)	• Implied	• N • Z	• SP -= 2; SP = & SP

Posr

Description	Supported addressing modes	This operation may change the flags	Operations
Pop SR (if SP > BP)	• Implied	• N • Z	• SP -= 1; SR = & SP

Push

Description	Supported addressing	This operation may	Operations

	modes	change the flags	
Push onto the stack	• Register access • Immediate		• Push R; SP += 2 • Push L; SP += 1

Read

Description	Supported addressing modes	This operation may change the flags	Operations
Make an IO synchronous device read request	• Memory related • Register access (get the address from registers)		

Req

Description	Supported addressing modes	This operation may change the flags	Operations
Make an IO device read request	• Memory related • Register access (get the address from registers)		

Reset

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Soft reset interrupt	• Implied		

Ret

Description	Supported addressing modes	This operation may change the flags	Operations
Return from a routine	• Implied		• Pop SR; Pop PC; SP = BP

Sbp

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations

Set the stack base in the MTU	• ALL MTUs		

Sehi

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Set the EMEM higher index in the MTU	• ALL MTUs		

Seli

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Set the EMEM lower index in the MTU	• ALL MTUs		

Seti

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Set interrupts flag	• Implied	• I	• I = 1

Sets

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Set system privileges flag	• Implied	• S	• S = 1

Sett

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Set timer interrupt flag	• Implied	• T	• T = 1

Shl

Description	Supported addressing modes	This operation may change the flags	Operations
Shift left register	• Register to register • Register to memory	• C (last bit out)	• Ra = Ra << Rb • R = R << M

Shr

Description	Supported addressing modes	This operation may change the flags	Operations
Shift right register	• Register to register • Register to memory	• C (last bit out)	• Ra = Ra >> Rb • R = R >> M

Sign

Description	Supported addressing modes	This operation may change the flags	Operations
Set N if the value is negative	• Register access • Memory related	• N	• N = R < 0 • N = L < 0 • N = M < 0

Sihi

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Set the IMEM higher index in the MTU	• All MTUs		

Sili

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Set the IMEM lower index in the MTU	• All MTUs		

Stbp

Description	Supported addressing modes	This operation may change the flags	Operations
Store BP in memory	• Memory related		• M = BP

Store

Description	Supported addressing modes	This operation may change the flags	Operations
Store the value of a register in memory	• Register to memory (invalid immediate) • Immediate to memory	• N • Z	• M = R

Stp

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Set the stack top in the MTU	• ALL MTUs		

Stsp

Description	Supported addressing modes	This operation may change the flags	Operations
Store SP in memory	• Memory related		• M = SP

Stsr

Description	Supported addressing modes	This operation may change the flags	Operations
Store SR in memory	• Memory related		• M = SR

Sub

Description	Supported addressing modes	This operation may change the flags	Operations
Subtract a value from a register	• Register to register	• N	• Ra = Ra - Rb

	• Register to memory	• O • Z • C	• R = R - M

Swap

Description	Supported addressing modes	This operation may change the flags	Operations
Swap two registers	• Register to register • Register to memory		• Ra <-> Rb

Syscall

Description	Supported addressing modes	This operation may change the flags	Operations
Call a system routine	• Implied		

Tbp

Description	Supported addressing modes	This operation may change the flags	Operations
Transfer stack base MTU into a register	• Register MTU	• N • Z	• R = MTU BP

Tehi

Description	Supported addressing modes	This operation may change the flags	Operations
Transfer EMEM higher index MTU into a register	• Register MTU	• N • Z	• R = MTU EMEM higher index

Teli

Description	Supported addressing modes	This operation may change the flags	Operations
Transfer EMEM lower index MTU into a register	• Register MTU	• N • Z	• R = MTU EMEM lower index

Tihi

Description	Supported addressing modes	This operation may change the flags	Operations
Transfer IMEM higher index MTU into a register	• Register MTU	• N • Z	• R = MTU IMEM higher index

Tili

Description	Supported addressing modes	This operation may change the flags	Operations
Transfer IMEM lower index MTU into a register	• Register MTU	• N • Z	• R = MTU IMEM lower index

Time

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Set the timer (16-bit milliseconds) and stop it.	• Register access (only full size 16-bit) • Memory related	• Z	• Timer = R • Timer = M

Tran

Description	Supported addressing modes	This operation may change the flags	Operations
Transfer data between registers	• Register to register	• N • Z	• Ra = Rb

Trsr

Description	Supported addressing modes	This operation may change the flags	Operations
Transfer SR into a register	• Register access (only low 8-bit)		• L = SR

Trt

Description	Supported addressing modes	This operation may change the flags	Operations
Transfer timer into a register	• Register access (only full size 16-bit)	• N • Z	• R = Timer

Tstart

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Start the internal timer	• Implied		

Tstop

Requires system privileges

Description	Supported addressing modes	This operation may change the flags	Operations
Stop the internal timer	• Implied		

Ttp

Description	Supported addressing modes	This operation may change the flags	Operations
Transfer MTU stack top into a register	• Register access (only full size 16-bit)	• N • Z	• R = MTU Stack Top

Write

Description	Supported addressing modes	This operation may change the flags	Operations
Make a IO write request	• Register to memory • Immediate to memory		

Xor

Description	Supported addressing	This operation may	Operations

	modes	change the flags	
Bit-wise xor between a register and a operand	• Register to register • Register to memory	• N • Z	• Ra = Ra xor Rb • R = R xor M