Atmel AVR instruction set

From Wikipedia, the free encyclopedia

The **Atmel AVR instruction set** is the machine language for the Atmel AVR, a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage.

Contents

Processor registers

There are 32 general-purpose 8-bit registers, R0–R31. All arithmetic and logic operations operate on those registers; only load and store instructions access RAM.

A limited number of instructions operate on 16-bit register pairs. The lower-numbered register of the pair holds the least significant bits and must be even-numbered. The last three register pairs are used as pointer registers for memory addressing. They are known as X (R27:R26), Y (R29:R28) and Z (R31:R30). Postincrement and predecrement addressing modes are supported on all three. Y and Z also support a six-bit positive displacement.

Instructions which allow an immediate value are limited to registers R16–R31 (8-bit operations) or to register pairs R25:R24–R31:R30 (16-bit operations ADIW and SBIW). Some variants of the MUL operation are limited to eight registers, R16 through R23.

Special purpose registers

In addition to these 32 general-purpose registers, the CPU has a few special-purpose registers:

- PC: 16- or 22-bit program counter
- SP: 8- or 16-bit stack pointer
- SREG: 8-bit status register
- RAMPX, RAMPY, RAMPZ, RAMPD and EIND: 8-bit segment registers that are prepended to 16-bit addresses in order to form 24-bit addresses; only available in parts with large address spaces.

Status bits

The status register bits are:

0. C Carry flag. This is a borrow flag on subtracts.
1. Z Zero flag. Set to 1 when an arithmetic result is zero.
2. N Negative flag. Set to a copy of the most significant bit of an arithmetic result.
3. V Overflow flag. Set in case of two's complement overflow.
4. S Sign flag. Unique to AVR, this is always N⊕V, and shows the true sign of a comparison.
5. H Half carry. This is an internal carry from additions and is used to support BCD arithmetic.
6. T Bit copy. Special bit load and bit store instructions use this bit.
7. I Interrupt flag. Set when interrupts are enabled.

Addressing

The following address spaces are available:

- The general purpose registers are addressed by their numbers (0–31), although the full 5-bit number is not stored in instructions that can only operate on a subset of those registers.
- I/O registers have a dedicated 6-bit address space, the lower half of which is bit-addressable; some parts have I/O registers outside this address space, which are called "extended I/O" and are only accessible as memory-mapped I/O in the data address space.
- The data address space maps the 32 general-purpose registers, all the I/O registers (including those also accessible through the I/O address space), and the RAM; it can be addressed either directly or indirectly through the X, Y and Z pointer registers, prepended if necessary by RAMPX, RAMPY and RAMPZ respectively.
- Program memory (flash) has a separate address space, addressed as 16-bit words for the purpose of fetching instructions
- For the purpose of fetching constant data, program memory is addressed bytewise through the Z pointer register, prepended if necessary by RAMPZ.
- The EEPROM is memory-mapped in some devices; in others, it is not directly addressable and is instead accessed through address, data and control I/O registers.
- The general purpose registers, the status register and some I/O registers are bit-addressable, with bit 0 being the least significant and bit 7 the most significant.

The first 64 I/O registers are accessible through both the I/O and the data address space. They have therefore two different addresses. These are usually written as "0x00 (0x20)" through "0x3F (0x5F)", where the first item is the I/O address and the second, in parentheses, the data address.

The special-purpose CPU registers, with the exception of PC, can be accessed as I/O registers. Some registers (RAMPX, RAMPY) may not be present on machines with less than 64 KiB of addressable memory.

Register	I/O address	Data address
SREG	0x3F	0x5F
SP	0x3E:0x3D	0x5E:0x5D
EIND	0x3C	0x5C
RAMPZ	0x3B	0x5B
RAMPY	0x3A	0x5A
RAMPX	0x39	0x59
RAMPD	0x38	0x58

A typical ATmega memory map may look like:

Data address	I/O address	Contents
0x0000 – 0x001F		Registers R0 – R31
0x0020 – 0x003F	0x00 – 0x1F	I/O registers (bit-addressable)
0x0040 – 0x005F	0x20 – 0x3F	I/O registers (not bit-addressable)
0x0060 – 0x00FF		Extended I/O registers (memory-mapped I/O only)
0x0100 – RAMEND		Internal SRAM

where RAMEND is the last RAM address. In parts lacking extended I/O the RAM would start at 0x0060.

Instruction timing

Arithmetic operations work on registers R0-R31 but not directly on RAM and take one clock cycle, except for multiplication and word-wide addition (ADIW and SBIW) which take two cycles.

RAM and I/O space can be accessed only by copying to or from registers. Indirect access (including optional postincrement, predecrement or constant displacement) is possible through registers X, Y, and Z. All accesses to RAM takes two clock cycles. Moving between registers and I/O is one cycle. Moving eight or sixteen bit data between registers or constant to register is also one cycle. Reading program memory (LPM) takes three cycles.

Instruction list

Instructions are one 16-bit word long, save for those including a 16-bit or 22-bit address, which take two words.

There are two types of conditional branches: jumps to address and skips. Conditional branches (BRxx) can test an ALU flag and jump to specified address. Skips (SBxx) test an arbitrary bit in a register or I/O and skip the next instruction if the test was true.

In the following:

- Rd is a register in the range R0-R31 or R16-R31 (depending on instruction)
- Rr is a register in the range R0-R31
- s is a bit number in the status register (0 = C, 1 = Z, etc., see the list above)
- b is a bit number in a general-purpose or I/O register (0 = least significant, 7 = most significant)
- K6 is a 6-bit immediate *unsigned* constant (range: 0–63)

- K8 is an 8-bit immediate constant; since it is used only in 8-bit operations, its signedness is irrelevant
- IO5 is a 5-bit I/O address covering the bit-addressable part of the I/O address space, i.e. the lower half (range: 0–31)
- IO6 is a 6-bit I/O address covering the full I/O address space (range: 0–63)
- D16 is a 16-bit data address covering 64 KiB; in parts with more than 64 KiB data space, the contents of the RAMPD segment register is prepended
- P22 is a 22-bit program address covering 2^{22} 16-bit words (i.e. 8 MiB)
- δD6 is a 6-bit *unsigned* displacement relative to the data address stored in the Y or Z pointer
- δP7 and δP12 are 7-bit (resp. 12-bit) *signed* displacements relative to the program address stored in the program counter

AVR instruction set

Arithmetic	Bit & Others	Transfer	Jump	Branch	Call
ADD Rd, Rr	BSET s	MOV Rd, Rr	RJMP δP12	CPSE Rd, Rr	RCALL δP12
ADC Rd, Rr	BCLR s	MOVW Rd+1:Rd, Rr+1:Rr	IJMP		ICALL
ADIW Rd+1:Rd, K6	SBI IO5, b		EIJMP	SBRC Rr, b	EICALL
	CBI IO5, b	IN Rd, IO6	JMP P22	SBRS Rr, b	CALL P22
SUB Rd, Rr	BST Rd, b	OUT IO6, Rr			
SUBI Rd, K8	BLD Rd, b			SBIC IO5, b	RET
SBC Rd, Rr		PUSH Rr		SBIS IO5, b	RETI
SBCI Rd, K8	NOP	POP Rr			
SBIW Rd+1:Rd, K6	BREAK			BRBC s, δP7	
	SLEEP	LDI Rd, K8		BRBS s, δP7	
INC Rd	WDR	LDS Rd, D16			
DEC Rd					
		LD Rd, X			
AND Rd, Rr		LD Rd, -X			
ANDI Rd, K8		LD Rd, X+			
OR Rd, Rr					
ORI Rd, K8		LDD Rd, Y+δD6			
EOR Rd, Rr		LD Rd, -Y			
		LD Rd, Y+			
COM Rd		LDD Rd, Z+δD6			
NEG Rd		LD Rd, -Z			
CP Rd, Rr		LD Rd, Z+			
CPC Rd, Rr					
CPI Rd, K8		STS D16, Rr			
SWAP Rd					
		ST X, Rr			
LSR Rd		ST -X, Rr			
ROR Rd		ST X+, Rr			
ASR Rd					
		STD Y+δD6, Rr			
MUL Rd, Rr		ST -Y, Rr			
MULS Rd, Rr		ST Y+, Rr			
MULSU Rd, Rr					
FMUL Rd, Rr		STD Z+δD6, Rr			
FMULS Rd, Rr		ST -Z, Rr			
FMULSU Rd, Rr		ST Z+, Rr			
		LPM			
		LPM Rd, Z			
		LPM Rd, Z+			
		ELPM			
		ELPM Rd, Z			
		ELPM Rd, Z+			
		SPM			

Instruction set inheritance

Not all instructions are implemented in all Atmel AVR controllers. This is the case of the instructions performing multiplications, extended loads/jumps/calls, long jumps, and power control.

Family	Members	Arithmetic	Branches	Transfers	Bit-Wise
Minimal Core	AT90S1200 ATtiny11 ATtiny12 ATtiny15 ATtiny28	ADD ADC SUB SUBI SBC SBCI AND ANDI OR ORI EOR COM NEG SBR CBR INC DEC TST CLR SER	RJMP RCALL RET RETI CPSE CP CPC CPI SBRC SBRS SBIC SBIS BRBS BRBC BREQ BRNE BRCS BRCC BRSH BRLO BRMI BRPL BRGE BRLT BRHS BRHC BRTS BRTC BRVS BRVC BRIE BRID	LD ST MOV LDI IN OUT LPM (not in AT90S1200)	SBI CBI LSL LSR ROL ROR ASR SWAP BSET BCLR BST BLD SEC CLC SEN CLN SEZ CLZ SEI CLI SES CLS SEV CLV SET CLT SEH CLH NOP SLEEP WDR
Classic Core up to 8K Program Space	AT90S2313 AT90S2323 ATtiny22 AT90S2333 AT90S2343 AT90S4414 AT90S4433 AT90S4434 AT90S8515 AT90C8534 AT90S8535 ATtiny26	new instructions: ADIW SBIW	new instructions: IJMP ICALL	new instructions: LD (now 9 modes) LDD LDS ST (9 modes) STD STS PUSH POP	(nothing new)
Classic Core with	ATmega103 ATmega603		new instructions:	new instructions:	

up to 128K	AT43USB320 AT76C711	(nothing new)	JMP CALL	ELPM	(nothing new)
Enhanced Core with up to 8K	ATmega8 ATmega83 ATmega85 ATmega8515	new instructions: MUL MULS MULSU FMUL FMULS FMULSU[1]	(nothing new)	new instructions: MOVW LPM (3 modes) SPM	(nothing new)
Enhanced Core with up to 128K	ATmega16 ATmega161 ATmega163 ATmega32 ATmega323 ATmega64 ATmega128 AT43USB355 AT94 (FPSLIC) AT90CAN series AT90PWM series ATmega48 ATmega88 ATmega168 ATmega162 ATtiny13 ATtiny25 ATtiny45 ATtiny85 ATtiny2313 ATmega164 ATmega324 ATmega328 ATmega644 ATmega165 ATmega169 ATmega325 ATmega3250 ATmega645 ATmega6450 ATmega406	(nothing new)	(nothing new)	(nothing new)	new instructions: BREAK
Enhanced Core with up to 4M	ATmega640 ATmega1280 ATmega1281 ATmega2560 ATmega2561	(nothing new)	new instructions: EIJMP EICALL	(nothing new)	(nothing new)
				new instructions: (from second	

XMEGA core	ATxmega series	new instructions: DES	(nothing new)	revision silicon - AU,B,C parts) XCH LAS LAC LAT	(nothing new)
Reduced Core	ATtiny10 ATtiny9 ATtiny5 ATtiny4	(Identical to minimal core, except for reduced CPU register set)	(Identical to classic core with up to 8K, except for reduced CPU register set)	Identical to classic core with up to 8K, with the following exceptions: LPM (removed) LDD (removed) STD (removed) LD (also accesses program memory) LDS (different bit pattern) STS (different bit pattern) Reduced CPU register set	(Identical to enhanced core with up to 128K, except for reduced CPU register set)

Instruction encoding

Bit assignments:

- rrrrr = Source register
- rrrr = Source register (R16-R31)
- rrr = Source register (R16-R23)
- RRRR = Source register pair (R0:R1 ... R30:R31)
- ddddd = Destination register
- dddd = Destination register (R16-R31)
- ddd = Destination register (R16-R23)
- DDDD = Destination register pair (R0:R1 ... R30:R31)
- pp = Register pair, W, X, Y or Z
- y = Y/Z register pair bit (0=Z, 1=Y)
- u = FMUL(S(U)) signed with 0=signed or 1=unsigned
- s = Store/load bit (0=load, 1=store)
- c = Call/jump (0=jump, 1=call)
- cy = With carry (0=without carry 1=with carry)
- e = Extend indirect jump/call address with EIND (0=0:Z, 1=EIND:Z)
- q = Extend program memory address with RAMPZ (0=0:Z, 1=RAMPZ:Z)
- aaaaaa = I/O space address
- aaaaa = I/O space address (first 32 only)
- bbb = Bit number
- B = Bit value
- kkkkkk = 6-bit unsigned constant
- KKKKKKKK = 8-bit constant

The Atmel AVR uses many split fields, where bits are not contiguous in the instruction word. The load/store with offset instructions are the most extreme example where a 6-bit offset is broken into three pieces.

Atmel AVR instruction set overview

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Instruction
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NOP
0	0	0	0	0	0	0	1	D D D D				R R R R				MOVW Rd,Rr Move register pair
0	0	0	0	0	0	1	0	d d d d				r r r r				MULS Rd,Rr
0	0	0	0	0	0	1	1	0	d d d			0	r r r			MULSU Rd,Rr
0	0	0	0	0	0	1	1	0	d d d			1	r r r			FMUL Rd,Rr
0	0	0	0	0	0	1	1	1	d d d			u	r r r			FMULS(U) Rd,Rr
0	0	0	$\overline{cy}$	0	1	r	d d d d d					r r r r				CPC/CP Rd,Rr
0	0	0	$\overline{cy}$	1	0	r	d d d d d					r r r r				SBC/SUB Rd,Rr
0	0	0	cy	1	1	r	d d d d d					r r r r				ADD/ADC Rd,Rr ROL/LSL Rd (ADC/ADD with Rd=Rr)
0	0	0	1	0	0	r	d d d d d					r r r r				CPSE Rd,Rr
0	0	1	0	0	0	r	d d d d d					r r r r				AND Rd,Rr
0	0	1	0	0	1	r	d d d d d					r r r r				EOR Rd,Rr
0	0	1	0	1	0	r	d d d d d					r r r r				OR Rd,Rr
0	0	1	0	1	1	r	d d d d d					r r r r				MOV Rd,Rr
0	0	1	1	K K K K				d d d d				K K K K				CPI Rd,K
0	1	0	$\overline{cy}$	K K K K				d d d d				K K K K				SBCI/SUBI Rd,K
0	1	1	0	K K K K				d d d d				K K K K				ORI Rd,K SBR Rd,K
0	1	1	1	K K K K				d d d d				K K K K				ANDI Rd,K CBR Rd,K
1	0	k	0	k k		s	d d d d d					y	k k k			LDD/STD through Z+k or Y+k
1	0	0	1	0	0	s	d d d d d					0	0	0	0	LDS rd,i/STS i,rd
16-Bit immediate SRAM-Address i																
1	0	0	1	0	0	s	d d d d d					y	0	0	1	LD/ST Rd through Z+/Y+
1	0	0	1	0	0	s	d d d d d					y	0	1	0	LD/ST Rd through −Z/−Y
1	0	0	1	0	0	0	d d d d d					0	1	q	0	LPM/ELPM Rd,Z
1	0	0	1	0	0	0	d d d d d					0	1	q	1	LPM/ELPM Rd,Z+
1	0	0	1	0	0	1	d d d d d					0	1	0	0	XCH Z,Rd
1	0	0	1	0	0	1	d d d d d					0	1	0	1	LAS Z,Rd
1	0	0	1	0	0	1	d d d d d					0	1	1	0	LAC Z,Rd
1	0	0	1	0	0	1	d d d d d					0	1	1	1	LAT Z,Rd
1	0	0	1	0	0	s	d d d d d					1	1	0	0	LD/ST Rd through X
1	0	0	1	0	0	s	d d d d d					1	1	0	1	LD/ST Rd through X+
1	0	0	1	0	0	s	d d d d d					1	1	1	0	LD/ST Rd through −X
1	0	0	1	0	0	s	d d d d d					1	1	1	1	POP/PUSH Rd
																1-operand instructions:
													0	0	0	COM

1	0	0	1	0	1	0	d d d d d					0	0	0	1	NEG
													0	1	0	SWAP
													0	1	1	INC
													1	0	1	ASR
													1	1	0	LSR
													1	1	1	ROR
1	0	0	1	0	1	0	0	$\bar{B}$	b b b			1	0	0	0	SEx/CLx Status register clear/set bit
1	0	0	1	0	1	0	1					1	0	0	0	Misc instructions:
								0	0	0	0					RET
								0	0	0	1					RETI
								1	0	0	0					SLEEP
								1	0	0	1					BREAK
								1	0	1	0					WDR
								1	1	0	q					LPM/ELPM
								1	1	1	0					SPM
								1	1	1	1					SPM Z+
1	0	0	1	0	1	0	c	0	0	0	e	1	0	0	1	Indirect jump/call to Z or EIND:Z
1	0	0	1	0	1	0	d d d d d					1	0	1	0	DEC Rd
1	0	0	1	0	1	0	0	k k k k				1	0	1	1	DES round k
1	0	0	1	0	1	0	k k k k k					1	1	c	k	JMP/CALL abs22
k k k k k k k k k k k k k k k k																
1	0	0	1	0	1	1	0	k k		p p		k k k k				ADIW Rp,uimm6
1	0	0	1	0	1	1	1	k k		p p		k k k k				SBIW Rp,uimm6
1	0	0	1	1	0	B	0	a a a a a					b b b			CBI/SBI a,b (IO-Operation)
1	0	0	1	1	0	B	1	a a a a a					b b b			SBIC/SBIS a,b (IO-Skip-Nextstep)
1	0	0	1	1	1	r	d d d d d					r r r r				MUL, unsigned: R1:R0 = Rr×Rd
1	0	1	1	s	a a		d d d d d					a a a a				IN/OUT to I/O space
1	1	0	c	12 bit signed offset												Relative jump/call to PC ± 2×simm12
1	1	1	0	K K K K				d d d d				K K K K				LDI Rd,K
1	1	1	1	0	$\bar{B}$	7-bit signed offset							b b b			Conditional branch on status register bit
1	1	1	1	1	0	s	d d d d d					0	b b b			BLD/BST register bit to STATUS.T
1	1	1	1	1	1	B	d d d d d					0	b b b			SBRC/SBRS skip if register bit equals B

References

1. Atmel. Application Note "AVR201: Using the AVR Hardware Multiplier" (http://www.atmel.com/Images/doc1631.pdf). 2002. quote: "The megaAVR is a series of new devices in the AVR RISC Microcontroller family that includes, among other new enhancements, a hardware multiplier."

External links

- GNU Development Environment (http://users.rcn.com/rneswold/avr/)
  - Programming the AVR microcontroller with GCC (http://www.linuxfocus.org/English/November2004/article352.shtml) by Guido Socher
  - A GNU Development Environment for the AVR Microcontroller (http://users.rcn.com/rneswold/avr/) by Rich Neswold
  - AVR Options (https://gcc.gnu.org/onlinedocs/gcc-3.3.5/gcc/AVR-Options.html) in GCC-AVR
- Atmel AVR instruction set PDF (151 pages) (http://www.atmel.com/images/Atmel-0856-AVR-Instruction-Set-Manual.pdf)
- AVR Instruction Set Simulator (ATmega32u4 for GCC Intel Hex Files) (http://starlo.org/blake/boardmicro/)
  - Google Play (https://play.google.com/store/apps/details?id=org.starlo.boardmicro)

The Wikibook *Embedded Systems* has a page on the topic of: ***Atmel AVR***

Retrieved from "https://en.wikipedia.org/w/index.php?title=Atmel_AVR_instruction_set&oldid=736946603"

Categories: Microcontrollers | Instruction set architectures
