Atmel AVR Brings RISC to 8-Bit World

Better Performance Than Other 8-Bit Chips With Same Low Cost

by Jim Turley

Proving the adage that all technologies eventually filter down into commodity products, Atmel has brought RISC design philosophy to 8-bit microcontrollers. Dubbed AVR, this new architecture provides all the usual benefits of RISC: faster clock rates, better performance, and more efficient compiler optimization. Atmel also promises better code density and lower cost than comparable 8-bit microcontrollers.

AVR competes with several well-established microcontroller dynasties such as the 6805, 68HC11, and 8051. Competition also comes from Microchip's PIC family, a more modern design that's expanded rapidly in the past few years. Atmel hopes AVR will appeal to embedded designers who are willing to tackle a new architecture to get more performance than the entrenched microcontroller families can provide.

AVR is the first in-house CPU design from Atmel, a billion-dollar company better known for its flash memory and E^2PROM products. The company also sells a dozen flash-based derivatives of the popular 8051 family, which it produces under license from Intel.

Design Melds RISC and Microcontroller Ideas

The CPU resembles most RISC processors but has smaller registers. It was originally developed by a pair of researchers in Trondheim, Norway, before their consultancy was acquired by Atmel in 1995. Core CPU development still takes place in Norway, while memory and peripheral development is centered in Atmel's San Jose (Calif.) facility.

The core features 32 identical 8-bit registers, as Figure 1 shows. Any register can hold addresses or data. Since 8-bit address pointers are fairly worthless even in an 8-bit device, the last six registers can be used in pairs, as address pointers.

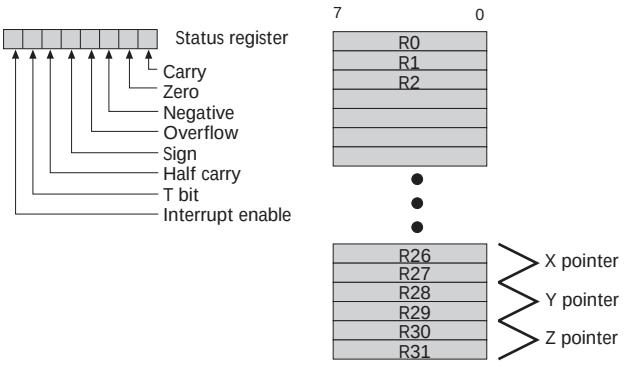

Figure 1. Unlike other 8-bit controllers, AVR has a set of 32 8-bit registers. The last six registers can be paired to form three address pointers.

Dubbed X, Y, and Z, these three meta-registers can be used for any load or store operation. The pointers can be post-incremented or predecremented at the programmer's option. Finally, a 6-bit displacement can be added to the contents of the pointer, a useful option for addressing array elements. This mode is not available for the X pointer; that opcode is reserved for the LDI (load immediate constant) instruction.

As with many low-end microcontrollers, the register file is mapped into the address space, and vice versa. The first 32 bytes of memory, 0x00–0x1F, correspond to registers R0–R31. The chip's status register—which contains the overflow, carry, sign, and other flags—as well as other "internal" registers are also memory mapped. This allows any register to be manipulated using standard memory references instead of special control-register instructions.

For all intents and purposes, the CPU has no pipeline. It retrieves both source operands, executes the instruction, and stores the result in a single clock cycle. Branch latency is one clock for taken branches. All operations are register-to-register; the chip follows a strict load/store model.

The great majority of AVR instructions are 16 bits long. Only four 32-bit instructions exist, allowing limited use of absolute 16-bit addressing. AVR separates the program and data spaces; although data pointers can be 16 bits, the PC (program counter) is only 12 bits wide, for 8K of code space.

Instruction Set As Regular As Possible

The compact instruction set necessarily forces some compromises, the first of which affects immediate values (literals). Very few instructions accept immediate values, and those that do (ADIW, SUBI, ORI, etc.) work only on the upper half (R16–R31) of the register set, as Table 1 shows. Even after shaving a bit from the operand-specifier field, these instructions sometimes have room for only 6-bit immediate values.

The ADIW and SBIW instructions (add/subtract immediate from word) are even more restrictive, operating on only the last eight registers, R24–R31. These instructions are meant primarily to add small offsets (0–63 bytes) to the X, Y, and Z pointers.

There is a wealth of conditional branch instructions: namely, two for each of the eight flags in the status register. With little 7-bit offsets, these instructions can deflect execution only 64 instructions in either direction. For bigger displacements, RJMP can shift code by 2K, which is usually plenty, given the chip's small code space.

AVR also has a collection of interesting "skip" operations (SBRC, SBRS, SBIC, and SBIS) that skip over the next instruction if any bit in any register is set or clear. If the skipped instruction is a long-displacement jump, these skips

Mnemonic	Description	Mnemonic	Description	Mnemonic	Description
Flow Control		**Bit Manipulation**		**Load/Store**	
JMP ◆	Jump absolute (24-bit)	SEC/CLC	Set/clear C flag (carry)	MOV	Copy register to register
RJMP	Branch relative (12-bit)	SEH/CLH	Set/clear H flag (half carry)	LD	Load indirect through X/Y/Z
IJMP ●	Jump indirect (Z)	SEN/CLN	Set/clear N flag (negative)	LD ●	Load indirect with postincremnt
RCALL	Call subroutine	SEZ/CLZ	Set/clear Z flag (zero)	LD ●	Load indirect with predecremnt
ICALL ●	Call subroutine indirect (Z)	SEI/CLI	Set/clear I flag (interrupt)	LDD ●	Load indirect with 6-bit offset
RET/RETI	Return/from interrupt	SES/CLS	Set/clear S flag (sign)	LDI	Load 8-bit immediate
CP/CPC	Compare/with carry	SEV/CLV	Set/clear V flag (overflow)	LDS ●	Load from 16-bit address
CPI	Compare with 8-bit immediate	SET/CLT	Set/clear T bit	LPS ●	Load from program space
CPSE	Compare, skip if equal	SBR/CBR	Set/clear bit in register	ST	Store indirect through X/Y/Z
SBRS/SBRC	Skip if register bit set/clear	BSET/BCLR	Set/clear bit in status register	ST ●	Store indirect with postincremnt
SBIS/SBIC	Skip if I/O bit set/clear	SER/CLR	Set/clear entire register	ST ●	Store indirect with predecremnt
BRcc	Conditional branch	SBI/CBI	Set/clear bit in I/O space	STD ●	Store indirect with 6-bit offset
Logical		**Arithmetic**		STS ●	Store to 16-bit address
AND	Logical AND	ADD/ADC	Add/with carry	IN/OUT	Input/output to/from I/O space
ANDI	Logical AND 8-bit immediate	ADIW ●	Add 6-bit immediate	PUSH/POP	Push/pop stack element
OR	Logical OR	SUB/SUBC	Subtract/with borrow	BLD/BST	Load/store T bit
ORI	Logical OR 8-bit immediate	SBIW ●	Subtract 6-bit immediate	**Miscellaneous**	
EOR	Logical exclusive-OR	SUBI/SBCI	Subtract 8-bit imm/w borrow	NOP	No operation
LSL/LSR	Logical shift left/right by 1 bit	INC/DEC	Increment/decrement register	SLEEP	Wait for interrupt
ROL/ROR	Rotate left/right by 1 bit	MUL ◆	Multiply $8 \times 8 \rightarrow 16$	WDR	Watchdog reset
ASR	Arithmetic shift right by 1 bit				
COM/NEG	One's/two's complement				
SWAP	Swap nibbles		Can use R16–R31 only	●	Not available on 90S1200, 1220
TST	Test for zero or minus		Can use R24–R31 only	◆	Future enhancement

Table 1. Atmel's AVR 8-bit RISC instruction set follows a strict load/store model, with a few simple indirect addressing modes, including postincrement and predecrement. The architecture also includes a number of individual bit-manipulation instructions.

can be used to effectively create conditional long-displacement branches. Alternatively, they can be used to skip a single arithmetic or logical operation in a string of operations, creating conditional operations somewhat similar to ARM's.

None of the AVR chips has a native multiply operation—much less a divide—although one has been defined. As defined, MUL multiplies any two 8-bit registers and deposits the 16-bit result in R0 and R1. When implemented, MUL executes in just two clock cycles, which is five times faster than the 68HC11's 10 clocks; even Motorola's newer 68HC12 (see MPR 5/27/96, p. 1) needs 3 clocks. Atmel expects to deploy its multiplier in future AVR chips as clock speeds increase and the chips take on simple signal-processing tasks.

Instructions Are Rich in Bit Manipulation

As with most microcontrollers, the AVR family has a host of bit-twiddling options, including 16 explicit instructions to set and clear every flag in its status register. This seems like a lopsided use of opcode space; the same result could have been achieved with normal logical operations. For deeply embedded applications, however, this was probably the right choice. Masking operations use precious address pointers and one or more registers; the SEx/CLx instructions use neither.

The chip can also set or clear any bit in any general-purpose or I/O register; SER and CLR wipe the contents of an entire register at once. SBR and CBR, which set or clear multiple bits at a time, are aliases for ORI and ANDI, respectively.

Initial Launch Includes Five Parts

Atmel launched its AVR product line with four basic chips: the 90S1200, the '2313, the '4414, and the '8515. The latter three devices are very similar, differing mainly in the amount of memory on the chip: 2K, 4K, or 8K of flash, with the amount of on-chip SRAM and E^{2}PROM also increasing.

The runt of the litter, the 1200, has only 1K of flash memory, no SRAM, no peripherals, and a restricted instruction set. With neither SRAM nor an external bus, the 1200 must use on-chip flash for data storage, which will slow execution considerably unless programmers can get by with juggling the register set alone. The 1200 is also the only chip in the family currently in production. In 1,000-unit quantities, the 20-pin 90S1200 sells for a paltry $1.65.

It's not often that the number of data bits outnumbers the pins on the package, but Atmel managed to get close with its 1220 device, an 8-pin version of the 1200. After power, ground, and crystal connections, only four pins are free for I/O. Most AVR chips come in 20-pin DIP or SOIC packages, which provide access to more I/O lines; only in a 40-pin package do the chips bond out their address and data buses for access to external memory.

All the parts are fabricated on Atmel's four 0.8-micron two-layer-metal fab lines in Colorado Springs and Rousset (France). This is the same memory process Atmel uses for its E^{2}PROM and flash devices, and for its 8051 chips with integrated flash. The 1200 measure about 24 mm^2 overall, and as

Price & Availability

Atmel's AT90S1200 is in production now. In 1,000-unit quantities, the part sells for $1.65; the 8-pin 1220 is less than $1. The 8515, with 8K of flash and 512 bytes of SRAM and E²PROM, is currently sampling; pricing has been set at $5.95 in 1,000-piece quantities. For more information, contact Atmel (San Jose, Calif.) at 408.487.2564 or access *www.atmel.com*.

the die photo in Figure 2 shows, the chip is nearly all logic. Memory processes typically don't produce very compact (or fast) logic, but most AVR chips will be dominated by memory and peripherals, and clock speeds aren't very high.

For Once, RISC Techniques Improve Code Density

It's sometimes hard to get excited about 8-bit processors, yet Atmel's AVR design is as different from others in its class as the first RISC machine was from big systems more than a decade ago. With its large register file and orthogonal instruction set, AVR is far more modern than its competitors.

Atmel's new CPU will be particularly appealing to programmers moving down the microprocessor food chain from 32-bit or 16-bit chips and who are accustomed to flexible register sets. For programmers moving up from, say, the 8051, AVR will be a real eye-opener.

For example, the 8051, 6805, and PIC all make do with a single accumulator; the 68HC11 and 'HC12 have just two. This makes AVR easier to program at the assembly level and easier to optimize with a compiler. The big register set reduces dependence on memory, which improves speed and shrinks data-storage requirements.

Counterintuitively, AVR's RISC-like instruction set also helps improve its code density over that of other 8-bitters, according to Atmel. Its CPI (compare immediate) instruction avoids the relatively awkward construct of loading, subtracting, and checking flags used on the 6805 and PIC. Adding two numbers on the 8051, 6805, or PIC usually involves shuffling both operands through the accumulator and storing the result; AVR simply adds two registers with one instruction in one cycle.

AVR is not pure RISC—some instructions are longer than others—nor is it the first 8-bit microcontroller with a big register file. Zilog's ever-popular Z8 has 16 banks of 16 registers and a very orthogonal instruction set. But at 5–15 clocks per instruction, the Z8 can't match AVR's speed.

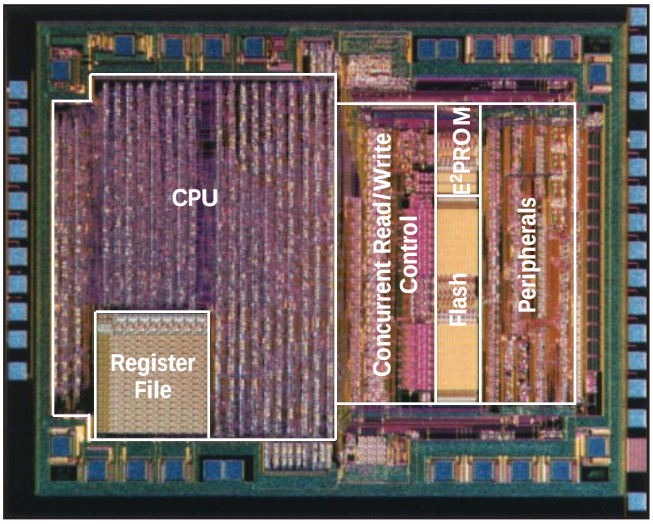

Figure 2. The 90S1200 measures about 4.3 × 5.5 mm in Atmel's 0.8-micron two-layer-metal flash-memory process.

Atmel Takes On Million Sellers

The 6805, 8051, and 68HC11 are the top-selling 8-bit families in the world, shipping millions of units every month. Motorola, for example, shipped its two-billionth 68HC05 in April. While AVR will not instantly overthrow these titans, it does provide substantial advantages over the entrenched leaders and blurs the line between 8- and 16-bit performance.

Although "high-performance" seems misplaced here, the AVR family should outperform other 8-bit microcontrollers and many 16-bit chips. At 20 MHz, its top clock rate is easily double that of other chips in its class. More important, almost all instructions execute in 1 or 2 clock cycles, versus 5–10 cycles for 8051, 6805, 68HC11, and PIC chips.

There is no lack of alternatives for the 8-bit user looking for more speed. Motorola's 68HC12 is a step up from the 'HC11; Philips and Intel are enticing 8051 users with the 8051XA (see MPR 10/3/94, p. 17) or the 251 family. Of these, Intel offers the smoothest upgrade path, with complete binary compatibility between the 8051 and the 251. Philips and Motorola both tout substantial size and speed advantages for users willing to reassemble (or recompile) their code.

The 'HC12, 8051XA, and 251 are more accurately 16-bit designs, with 16-bit internal data paths and 16-bit arithmetic operations, but they still require three or more clocks for even the simplest calculations and most basic register operations. Moreover, the clock rates of these parts are not substantially faster than Atmel's and can't make up for inherently inefficient execution.

In short, AVR offers 16-bit performance at an 8-bit price. For users who want on-chip memory but don't need 16-bit data types or binary compatibility with a previous generation, Atmel offers better price/performance for designers willing to let go of the older families' apron strings.

With only one part in production, it's far too early to tell whether Atmel's new family will make a dent in the marketplace. But at less than $1 in volume, the tiny 1220 should certainly be attractive to designers of toys, control systems, and consumer items. The cost of 8-bit microcontrollers is historically determined by their peripherals, and with none to speak of, the 1200 is probably not a good indicator of future pricing. As Atmel rolls out the rest of the family this year, the effect of its bold move will become clearer. M