Comparing two CPU architectures: AVR and ARM

By: Adar Guy
Contact: adarguy10@gmail.com

Computers perform very basic commands such as reading/writing data, and calculating basic arithmatic with low-level commands. A complete set of commands is known as the computer's instruction set and most higher-level programming languages such as Java, Python and C++ use compilers that convert the complicated code into these simple commands. Since the beginning of computers, CPU architects have worked endlessly to produce more efficient designs, trading chip space for a larger data bus. This has sparked a debate between computer scientists and has led to the creation of many different design implementations.

The Atmel AVR is an 8-bit microcontroller that uses a modified Harvard architecture This architectural design is based on the original Harvard archtecture of physically separating program and data memory but the CPU executes from the cache. The ARM has a 16 and 32 bit architecture and follows the von Neumann design of a single storage memory system for both data and program memory. Both these architectures rely on a RISC (reduced instruction set computing) CPU design which is based on the idea that a reduced or simplified instruction set, combined with a microprocessor that is capable of carrying out efficient commands will provide higher performance than those using more versatile architectures.

The Atmel AVR contains 32 single-byte registers from r0 to r31. They are all general purpose but some have intentions behind them. Registers 26-27, 28-29, and 30-31 are paired registers to be used as 16-bit pointer registers. The first 16 registers 0-15 cannot operate on a register and an immediate value so these operations are saved for the higher value registers (a design flaw). The ARM microcontroller has 37 registers each that is 32-bits long but is sectioned into 1 program counter register, 1 current program status register, 5 saved program status registers, and 30 general purpose registers. These groups are arranged and become accessible with each processor mode. These processor modes can access any set of registers r0-r12, r13 (stack pointer), r14 (subroutine link register that stores branch results), r15 (program counter) and the current/saved program status registers.

The AVR and ARM architectures both use very similar assembly instructions to perform the same calculations. In terms of the process, ARM can hold larger values in each register than AVR so it can perform larger calculations more efficiently.

The syntax for adding two constants together in the AVR assembly language is:

```
LDI r16, 0x36
LDI r17, 0x42
ADD r16, r17
```

The syntax for adding two constants together in the ARM assembly language is:

```
LDR r16, =0x36
LDR r17, =0x42
ADD r16, r17
```

Works Cited

Advantages of ARM over AVR. Society of Robots, 2005. Web. 23 Mar. 2015. <http://www.societyofrobots.com>.

Atmel Corporation. Atmel Corporation, 2014. Web. 21 Mar. 2015. <http://www.avrfreaks.net>.

avr-asm-tutorial.net. *Introduction to AVR assembler programming for beginners*. Ed. Gerhard Schmidt. Darmstadt, Feb. 2001. Web. 22 Mar. 2015. <http://www.avr-asm-tutorial.net/avr_en/beginner/REGISTER.html>.

"The Arm Instruction Set." *Simple Machines*. Arm Holdings, Nov. 1990. Web. 23 Mar. 2015. <http://simplemachines.it/doc/arm_inst.pdf>.

Vipin Hakeem. *Digital Design and Programming*. My Free Copyright, Sept. 2010. Web. 22 Mar. 2015. <http://v-codes.blogspot.ca/2010/09/difference-between-harward-and-von.html>.