What is a Flip-Flop and a Register?

You can think of a D flip-flop as a one-bit memory. The something to remember on the D input of flip-flop is remembered at the positive edge of the clock pulse.

D	Q
0	0
1	1

Something to Remember — Remember Me! — What I Remember

A register is a collection of flip-flops sharing the same clock input.

Source: https://toolbox.eia.edu/Electronics%20Basics.html

ATMega328P Functional Diagram

Figure 5-2. ATmega328P Block Diagram, adapted from: ATMEL. (2015).

Source: ATmega328 Data Sheet, Chapter 5 "AVR CPU Core" https://www.microchip.com/wwwproducts/en/ATmega328p

The AVR Engine

Let's adopt the analogy used by Charles Babbage when he talked about Computers as a working Engine. For closer look see the article in Wired and ATmega328 Wikipedia page.

Photo credit Maxwell Hamilton, Flickr

Instruction Set Architecture

- The Instruction Set Architecture (ISA) of a microprocessor includes all the registers that are accessible to the programmer, in other words, registers that can be read/written by the instruction set of the processor.
- With respect to the AVR CPU illustrated in Figure 5-2, those ISA registers include the 32 x 8-bit general purpose registers, status register (SREG), the stack pointer (SP), and the program counter (PC).

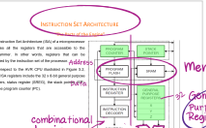


Figure 5-2. AVR CPU Processing Unit (SA Diagram)

Source: ATmega328 Data Sheet, https://www.microchip.com/wwwproducts/en/ATmega328p

AVR CPU Core Architecture

"Features of the Engine"

Part I

- Reduced Instruction Set Computer (RISC): The instruction set of the computer is carefully and completely designed to connect efficiently the operations of both. In this way, a relatively high performance at processor

CISC vs RISC

	1975+	2004
	CISC	RISC
	8051 Microcontroller	ATmega328P Microcontroller
Clock	12 MHz	16 MHz
Instruction	1 to 5 cycles	1 cycle

- Mostly 16-bit fixed-length instructions: Instructions have from zero to two operands. Many of today's RISC microprocessors have up to three operands.
- The Register File: all the AVR CPU contains 32 x 8 bit general purpose working registers (or Gprs). **General Purpose Registers**: ATMEGA328P has 32 registers; therefore, identifying each register requires 5 bits.
- Load-store memory access: Before you can do anything to data, you must first load it from memory into one of the general purpose registers. You then can operate on the registers and finally store the data back into the data. Finally, you store your answer back into memory.

r16 + r16 =

load Do Something store

Accumulator

Scratch Pad Memory

AVR CPU Core Architecture

"Features of the Engine"

Part II

- Modified Harvard memory model: A Harvard memory model separates Program and Data memory into separate physical memory systems (Flash and SRAM) that appear in different address spaces. A Modified Harvard memory model has the ability to read/write data items from/to program memory using special instructions. A Princeton memory model computer has a single address space, shared by both the program and data.
- A Two-stage Instruction Pipeline (fetch and execute) resulting in most instructions being executed in one clock cycle. Consequently, the performance of a 20 MHz processor would approach 20 MIPS (Millions of Instructions Per Second). Comparison this with the 8051 Complex Instructions Set Computer (CISC) computer which takes a minimum of 12 clock cycles to execute a single instructions (12 MHz clock = 1 MIPS).
- Simplicity of the computer architecture translates to a faster learning curve and utilization of the machine by the student.

AVR CPU Instructions

"The Language of the Machine"

The instruction list of our AVR CPU can be functionally divided into classified into:

1. Data Transfer
2. Arithmetic and Logical
3. Bit and Bit-test
4. Control Transfer (Program Sequencing)
5. MCU Control

load Store

- Data Transfer instructions are used to Load and Store data to the various Register Repository, also known as the Register File.
  - Exceptions are the push and pop instructions which modify the Stack Pointer.
  - By definition these instructions do not modify the status register (SREG).
- Arithmetic and Logic Instructions: plus 50 and 84 Text Instructions are the "ALU" in operate on the data contained in the general purpose registers.
  - Flags contained in the Status Register (SREG) provide important information concerning the results of these operations.
  - For example, if you are adding two signed numbers together, you will want to know if the answer is correct. The state of the overflow flag (V) in the SREG gives you the answer to the question ("1 = error, 0 = error).
- As the AVR processor fetches and executes instructions it automatically increments the program counter (PC) so it always points to the next instruction to be executed. Control Transfer Instructions allow you to change the contents of the PC when conditions are satisfied.
  - Continuing our example if an error results from adding two signed numbers together we may want to conditionally JUMP to branch to an error handling routine.

Instruction Fetch and Execute

"The Basic Cycles of The Engine"

Down both, our computer lives by the Fetch and Execute instructions, the heart and brother of the computer programmer. For this reason, the programmer views the computer as a vehicle for executing a set of instructions. This perspective is confined by the instruction set architecture (ISA) of the computer.

2 state FSM

sequential logic

control

Datapath

Figure 5. The Two Basic Stages of All Microprocessor Instruction Execution

Figure 6. AVR CPU Registers and logic used to fetch and decode an instruction

Source: ATmega328 Data Sheet, https://www.microchip.com/wwwproducts/en/ATmega328p

Advanced Version: Princeton Memory Model Instruction Fetch Cycle

1. The CPU places the current value of the program counter (PC) on the address bus and sets the read control line.
2. The RAM/ROM memory module, via the address of the instruction and presents the value on the data bus.
3. The value from the data bus is stored into the instruction register and the CPU clears the read control line. The instruction register now holds the next action to be executed.
4. The program counter is incremented so it points to the next instruction to be executed.
5. The instruction decoder interprets and implements (executes) the instruction.

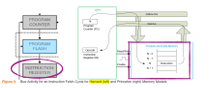


Figure 6. The Activity for an Instruction Fetch Cycle for Harvard (left) and Princeton (right) Memory Models

I/O Address Space versus Memory Mapped I/O

- Input and Output ports have traditionally been treated as separate parts of the computer.
- The AVR includes an `in` instruction to read from an I/O port and an `out` instruction to write to an I/O port.
- The AVR has 64 I/O registers accessible to these two instructions.

Problem: The Atmel ATmega line of Microcontrollers needs more than 64 I/O registers (SPIFI, Timers, ...)

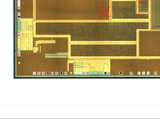
Solution: Instead of booting at computers having 5 basic elements (Input, Output, ALU, CPU, Memory), you can simplify the design to only three (CPU, ALU, and Memory) new allowing the CPU to access 160 "extended" I/O registers using RAM instructions like `lds` (load from SRAM) and `sts` (store to SRAM).

- This was such a powerful technique that Atmel extended the I/O mapping to include the 32 general purpose registers, the original 64 I/O registers, and the 160 extended I/O registers. The overlaying of the I/O address space with the SRAM address space is shown in the next slide.
- A side benefit of the double mapping is the logic number of ways of accessing data within SRAM (addressing modes) versus the limited number of instructions and addressing modes available for accessing the original 64 I/O registers (i.e., `in`, `out`).
- It is very important to realize that I/O registers are not contiguous within the address space (I/O or SRAM). The mapping is simply a convenient way of accessing registers physically located in diverse locations within the Silicon chip.

ATmega Harvard Princeton Memory Models

Source: ATmega328P Data Sheet http://www.atmel.com/images/atmel-42735-8-bit-avr-microcontroller-atmega328-328p_datasheet_complete.pdf Chapter 7 and Memory Figure 8-1

ATmega328P I/O Memory Map[1]



Source: ATmega328P Data Sheet http://www.atmel.com/images/atmel-42735-8-bit-avr-microcontroller-atmega328-328p_datasheet_complete.pdf Chapter 28 Register Summary

Appendix A Processor Control and Datapath[2]

Control	Datapath
Component of the processor that commands the datapath, memory, data, I/O devices according to the instructions of the memory	Components of the processor that perform arithmetic operations and holds data

[2] https://www.dcs.bbk.ac.uk/~roman/H19/H19-Lecture2.pdf

1822 – Charles Babbage and the Analytical Engine

- Conceived by Babbage, the engine established the basic principles upon which modern general-purpose digital computers are constructed. The **mechanical** machine performed instructions dictated by a punched course, with the variable values being determined by a second set of punched cards. The machine was designed to have a memory capacity of 1,000 numbers of 50 decimal digits each (about 21 kB).
- Neither the Analytical Engine or Difference Engine (1830), a second more complex design for a programmable mechanical device (in 19 x 6 ft), were ever completed by Babbage. Both machines were "the difference engine" and "the analytical engine" in the machine room behind me.

1843 – Ada Byron and the First Computer Program –

- Ada Byron, Lady Lovelace, was one of the most profound characters in computer history. Augusta Ada Byron was born December 10, 1815 the daughter of the flamboyant poet Lord Byron. Ada was brought up to be a mathematician and at the ill age of six (almost quite of 6) Ada studied that she wanted to be a mathematician. Ada's mother, Lady Byron, was a mathematician and was separated from Lord Byron shortly after the birth of Ada. Lady Byron was the subject of many articles attacking Babbage's Analytical Engine. Lady Lovelace prepared a translation of a memoir on the Analytical Engine by the Italian military engineer Luigi Menabrea. Ada added her own notes to the memoir, almost doubling its size. Her notes are important in the early history of computers. She described a method for calculating Bernoulli numbers using the Analytical Engine, and this is recognized by some historians as the world's first computer program. Charles Babbage also wrote 93 recollections for more than a century, many of which are not to be found anywhere else. Additionally the Babbage Archive Project is an online resource that aims to digitize all of Babbage's papers and publish them online. Source: http://www.fi.edu/learn/sci-tech/babbage/babbage.html?fcfi=3

1937 – Howard Aiken's

- Howard Hathaway Aiken at **Harvard** proposed in 1936 the idea for an Automatic Sequence Controlled Calculator — this was to be by the large scale calculator. This similar to the Analytical Engine, the machine used a combination of **electromechanical** devices, including punch cards, and was built by IBM. The Mark I was 51 feet long, 8 feet high, and weighed 5 tons. It had 72 accumulators that could perform 23 decimal digit additions or subtractions every three seconds, and had the mechanical capability of 1 ms per multiply. Output was stored by IBM punches or by typewriters. Aiken's machine was a decimal machine, which means that it represented numbers as decimal digits rather than as binary numbers. Babbage's machine was also decimal. Aiken was unaware of Babbage's work when he began his project. Aiken's machine was controlled by a sequence of instructions that were read from a paper tape. Source: http://www.fi.edu/learn/sci-tech/babbage/babbage.html?fcfi=3

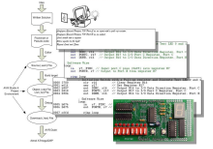
1945 – Dr. John Von Neumann and the Electronic Discrete Variable Computer (EDVAC)

- Von Neumann was involved in the design of the Electronic Discrete Variable Computer (EDVAC) in 1945. With after the ENIAC Plus it was one of the first operational stored-program computers. The technical team was in the Moore School of Electrical Engineering, including Von Neumann, Presper Eckert, Jr. (Inventor) John Mauchly, and John Brainerd. Source: http://www.fi.edu/learn/sci-tech/babbage/babbage.html?fcfi=3

- This computer was the "brainchild" of John von Neumann, who was a consultant to the ENIAC team. With after the EDVAC Plus it was one of the first operational stored-program computers. The technical team was in the Moore School of Electrical Engineering, including Von Neumann, Presper Eckert, Jr. (Inventor) John Mauchly, and John Brainerd. Source: http://www.fi.edu/learn/sci-tech/babbage/babbage.html?fcfi=3

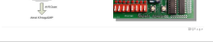
1947 – The First Case Bug

- Grace Hopper, working on the Harvard Mark I computer, found a moth stuck in one of the components. They taped the moth in their logbook and declared it "first actual case of bug being found". The words "bug" and "debug" now become a standard part of the language of computer programmers.



1951 when Grace Hopper and Remington Rand Corporation's Eckert-Mauchly Division

- Developed its compiler, A-0 System, which translates a pseudocoded code of the general concepts of parallel branch, storage/program computers. That it is used between binary machine or symbolic coding. 1957 – The symbolic system FORTRAN



Classic Microcontroller Architecture

The AVR is divided into a maximum subsystem elements as shown in the figure 3-1. The functional blocks coordinate each logic and maintains the instructions held in the instruction register have directed from the control signals needed to execute the instruction. The three path connects the Harvard Program Registers with the Program Memory and the Instruction Register and Decoder. The Program Memory contains the instructions loaded from the program memory.

The development of the program and data memory described by the Wayback, is also known as the **Princeton** memory model. The architecture of our AVR processor separates these two types of memory by the Harvard Program Memory and Data Memory Modified. The organization of program and data memory more resembles the Princeton/Atlas's computer, than the EDSAC computer, and is therefore known as the **Harvard** memory model.

The input and output functions of Figure 3-1 will be covered together and mostly called input/output (I/O). For more sensitivity, the Intel 8051 includes all the on-chip (Wayback's) I/O functions are a supplementary coprocessor model of microcontroller, it's not even the ATmega328.

Figure 3-1 shows the Princeton architecture is the counterpoint to the Harvard. Note: Flash Program Memory, SRAM Data Memory, Control Unit, Datapath, and Input/Output.

Figure 3-1 Basic Microcontroller Architectural Elements

Appendix G Microprocessor versus Microcontroller

	Typical Microprocessor	Low-Cost Microcontrollers
Core	Expensive ($100s)	Cheap (<$10)
	Packed all in one DIP	
Speed	3 GHz (glue - SIP)	20 MHz (cheap – SIP) effective = 20 MIPS
	Packed all in hundred DIP)	
Cores	Up to Four	One
Pipeline Stages	4 to 35	4 to 5
Address Bus	64 bits (2^64 = 2^27)	16 bits (2^16 = 2^27)
	2^64 = 18,446,744,073,709,551,616 = 10^19	2^16 = 65,536 = 10^5
	Factor of approximately a Quadrillion (10^15)	
Data Bus	32 bits to 64 bits	8 bits
Instruction Set	Complex	Simple and CISC-oriented (Limited)
MCU	Floating Point Unit	Simple ALU (+, -, & plus logic operations)
Program & Data Memory	Via	Yes
Peripherals	Via	Parallel I/O, Counter/Timers, ...

Appendix H Two-stage Instruction Pipeline

Pipelining A technique that breaks down, and, to enhance processing or time. Instructions, into smaller distinct stages or become independently so that multiple instructions can be processed concurrently.

From the Harvard/Princeton/Von Neumann/Wayback/AVR/Pic/Intel, Architects and authors experts in Figure 4-1 Block Diagram of the AVR Architecture.

To order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data instructions in the program memory can execute with a single instruction fetching. While one instruction is being executed, the next word in the bus is fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is in System Reprogrammable Flash memory.

A platform to begin and end with a register, consolidating a cycle. Between the registers is combinational logic. Although modern chipsets, Flash program memory can be erased as combinational logic with an address generating a result (Data). With respect to our AVR architecture the figure 4-2 the two registers of the pipeline are the Instruction Register and the Instruction Decoder. The Program Counter is controlled by the internal control (PC) that keeps track of the current instruction in execution, computer operation (program flow), and the status (SR) flag that defines the current CPU operation.

Figure 10 Five Level Execution Cycle of the Advanced Virtual RISC microcontroller

Instruction	Fetch	Instruction	Execute
Instruction	Fetch	Instruction	Execute
	Instruction	Fetch	

Time	T1	T2	T3	T4	T5	T6
Instruction	Fetch	Instruction	Execute			
		Instruction	Fetch	Instruction	Execute	

For most instructions, especially one based on a modified Harvard memory model, program memory is not accessed during the execution cycle. This memory therefore could be used to fetch the next instruction to be executed, in parallel with the execution unit of the current instruction. Now there is an opportunity for pipelining, as shown in Figure 10. While one instruction is being executed, the next instruction is being fetched from memory. As shown in Figure 10, with the second stage of executing the instruction. The two stage instruction pipeline can be called instruction pointer and formed in some of the earliest microprocessors including the microAVR.

Figure 11 Instruction Pipeline of the microAVR Microprocessor

Time	T1	T2	T3	T4	T5	T6
Instruction	Fetch	Instruction	Execute			
		Instruction	Fetch	Instruction	Execute	

For the RISC architecture most instructions are executed in a single cycle (instruction via internal instructions). Unlike pointer and branches all instructions take one cycle to fetch and one cycle to execute, after or initial delay of one cycle to fill the pipeline, known as latency. Each instruction will take only one cycle to complete.

Figure 12 Program Execution in a RISC/VLIW, Two-Stage Instruction Pipelined Architecture

Time

	1	2	3	4
Stack	Instr. 1	Instr. 2	Instr. 3	Instr. 4
Execute		Instr. 1	Instr. 2	Instr. 3

Hoppelhopp or over the clouds drops simulation with implementing the pipeline for example the Strictly, and when computing access, our performance would be better that shown with pipelined design.

Anatomy of an Assembly Instruction

Sample Code Segment

Machine Code	Assembly Code
Address	Data
0000 0000 0000 0000	0x2709	clr r16
0000 0000 0000 0001	0x2711	clr r17
1111 1101 0000 0111	0xFD07	sbic PINC, 7
1111 1101 0101 0101	0xFD55	sbic PINC, 5

- The **Operation Code** or **Opcode** for short, is a mnemonic that tells the CPU what instruction is to be executed. In the sample code above that would be `clr` (clear register), `sbic` (skip if bit in I/O location), One or more operands follow the Opcode.
- The **Operand(s)** specify the location of the data that is to be operated on by the CPU. In many cases it is the Arithmetic Logic Unit (ALU) that performs the specified operation.

Development Steps