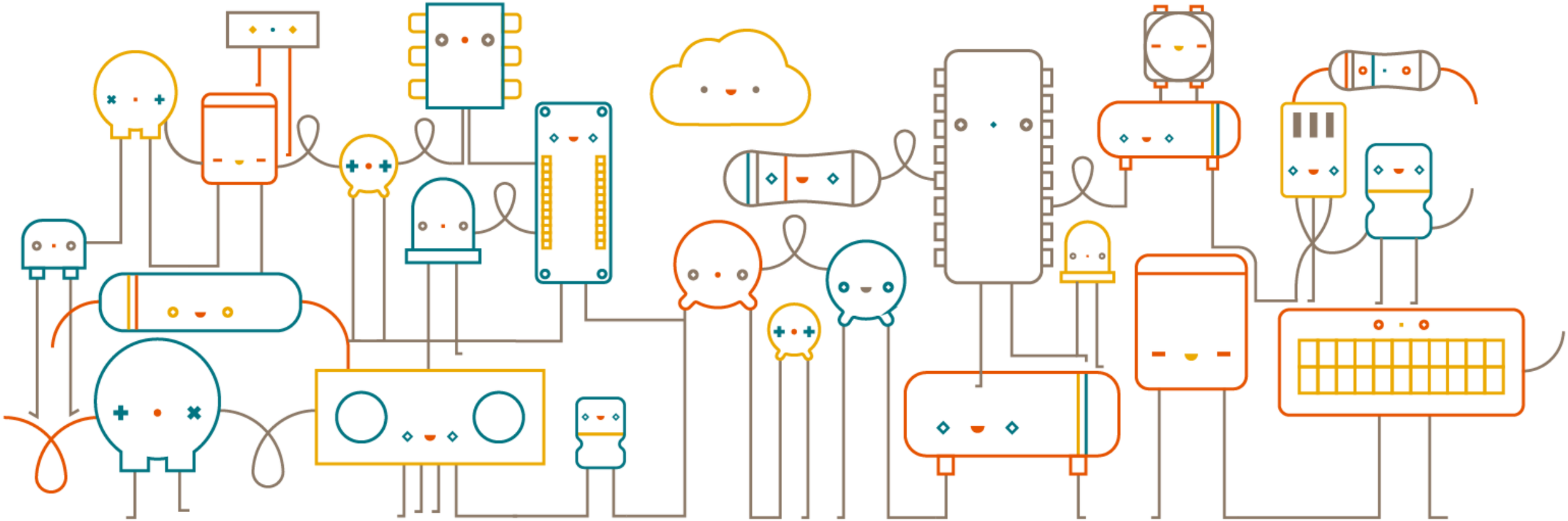


Reversing & Forensic with Arduino: a quick practical guide

Cesare Pizzi

ARDUINO DAY 2019

March 16TH
day.arduino.cc
#ArduinoD19

About Me

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: general thoughts

What happens when a sketch is loaded into the Arduino board (or in general to an AVR MCU)?

Do we have any way to understand what is going on , even if we don‘t have the source code?

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: general thoughts

Reverse engineering? Is that legal? Why should I do it?

We know we are in a gray area...
Article 6 (2009/24/EC)

Decompilation

1. The authorization of the right holder shall not be required where reproduction of the code and translation of its form within the meaning of points (a) and (b) of Article 4(1) are indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs, provided that the following conditions are met:

(a) those acts are performed by the licensee or by another person having a right to use a copy of a program, or on their behalf by a person authorized to do so;

(b) the information necessary to achieve interoperability has not previously been readily available to the persons referred to in point (a); and

(c) those acts are confined to the parts of the original program which are necessary in order to achieve interoperability.

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: general thoughts

SORINT.lab

From https://it.wikipedia.org/wiki/Reverse_engineering:

Nel caso specifico italiano, la reingegnerizzazione a scopo di interoperabilità con altri sistemi (e solo a questo scopo) è un atto pienamente lecito ai sensi dell'art. 64 della legge 633 del 22 aprile 1941, come modificata dall'art. 5 del D. Lgs. 518/1992, sia in senso "leggero" (qualora egli compia tali atti durante operazioni di caricamento, visualizzazione, esecuzione, trasmissione o memorizzazione del programma che egli ha il diritto di eseguire) che in senso di decompilazione vera e propria, ma solo al fine di permettere l'interoperabilità del software con altri programmi.
L'accezione di software è estesa per analogia a concetti informatici quali il formato di un file o la struttura interna di un protocollo.

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: general thoughts

- AVR is not only Arduino. It is used in a lot of different products: home automation, industrial PCL, HID devices (Xbox controllers for example with AT43USB353M), Amazon Dash Button
- AVR family product is very wide, with a lot of different chips with different features
- Atmel (now Microchip) did a great job in documenting all the features: if lost, just look for data sheets on the website.

March 16TH
day.arduino.cc
#ArduinoD19

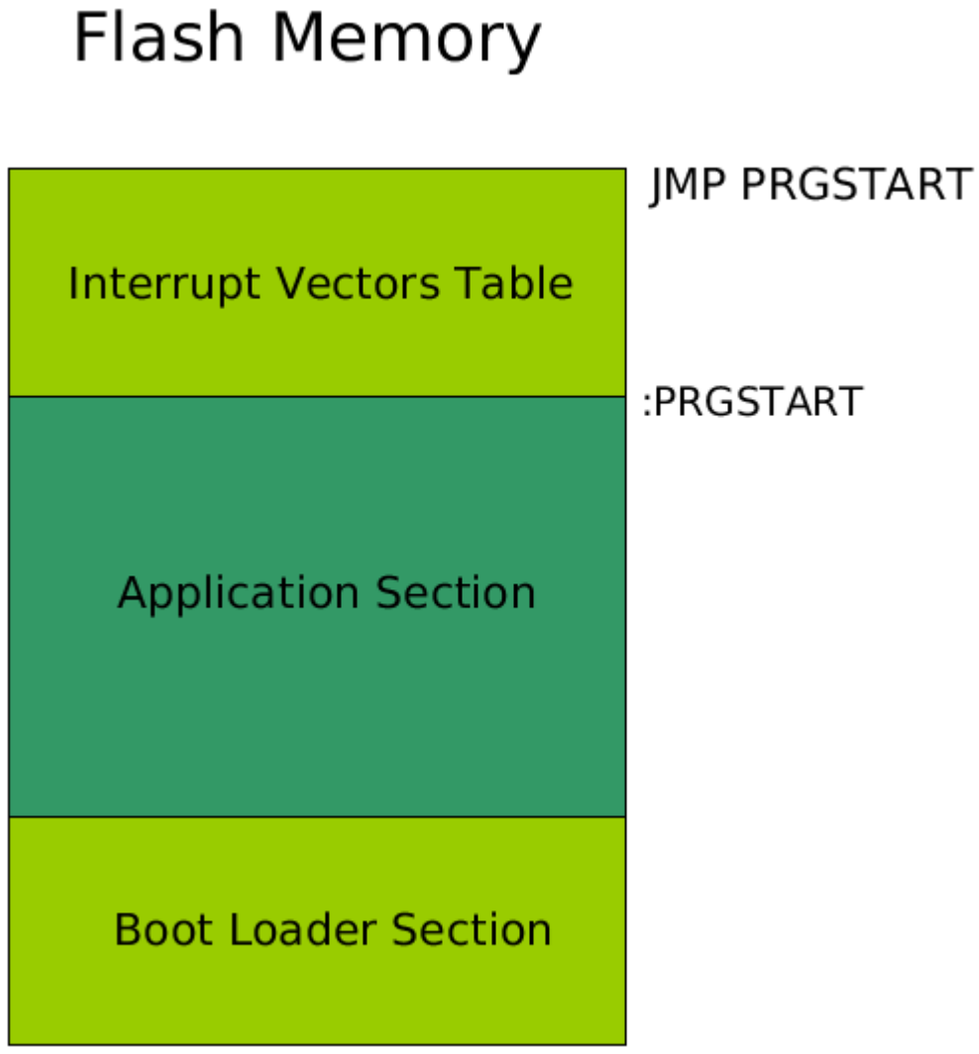
Arduino RE: Arduino/AVR architecture

SORINT.lab

- Flash, EEPROM and SRAM are all integrated on the same chip
- Program instructions are stored in the Flash
- Size of the programs can vary from 32 to 64 KB (or 256KB), depending on the MCU model. All the running code must reside on the flash
- EEPROM is available for permanent data storing

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: Arduino/AVR architecture



March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: steps

- Dump flash memory
- Analysis level 0: visual inspection
- Analysis level 1: use well known forensics utilities
- Analysis level 2: what about debugging
- Analysis level 3: when the going gets tough…keep calm and start the disassembler

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: dump flash

The dump of the MCU flash is done via the well-known AVR toolchain utilities:

avrdude -C/opt/arduino-1.8.2/hardware/tools/avr/etc/avrdude.conf -q
-patmega328p -carduino -P/dev/ttyACM0 -b115200 -D -Uflash:r:/tmp/flash.hex:i

-D disable auto erase
-q quiet output
-p part no (MCU)
-c programmer
-U perform memory operation: read(r):filename:format (i=HEX) (r=raw bin)

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: dump flash

If the USB connection is not available (because the bootloader is not present or there is no USB connection at all), we have some alternatives:

- Chip removal
- Arduino to Arduino connection
- JTAG or similar connection

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: dump flash

```
[+] Running for MCU: atmega328p with insert-flag: 1092C100
:20 0000 00 D2C00000FEC00000FCC00000FAC00000F8C00000F6C00000F4C00000F2C00000 46
:20 0020 00 F0C00000EEC00000ECC00000EAC00000E8C00000E6C00000E4C00000E2C00000 78
:20 0040 00 E0C00000ABC20000DCC00000DAC00000D8C00000D6C00000D4C00000F6C10000 E4
:20 0060 00 D0C0000065C200003AC20000CAC00000C8C00000C6C00000C4C00000C2C00000 2F
:20 0080 00 C0C00000BEC00000BCC00000BAC00000B8C00000B6C00000B4C00000B2C00000 98
:20 00A0 00 B0C00000AEC00000ACC00000AAC00000A8C00000A6C00000A4C00000A2C00000 F8
:20 00C0 00 A0C000009EC000009CC000009AC0000098C0000096C0000094C0000092C00000 58
:20 00E0 00 90C0000000022002500280002B002E00310034000201000005010801 0B010000 65
:20 0100 00 2100240027002A002D00300033000101000004010701 0A01050505050705 0808 6F
```

:	Line start
20	Byte count (on the line)
0000	Address
00	Record type (00 Data, 01 EOF, …).
ABCD	Data
78	Checksum (two's complement of the least significant byte of the sum of all decoded byte values in the record preceding the checksum

ARDUINO DAY 2019

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: level 0 – Visual Inspection

SORINT.lab

Tools used:

→ strings
→ less
→ other basic *nix commands

DEMO

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: level 1 – Forensics Utilities

SORINT.lab

Tools used:

→ xorstring
→ xorsearch
→ floss

DEMO

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: level 2 – Debugging

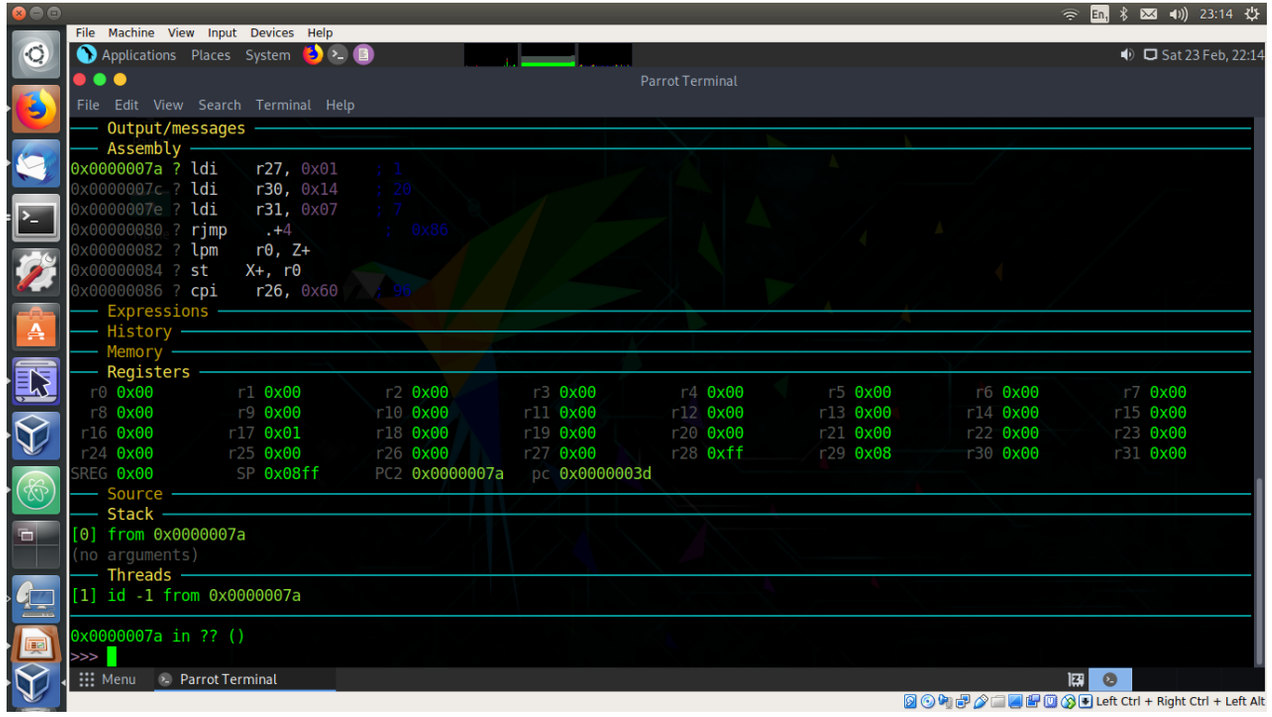
SORINT lab

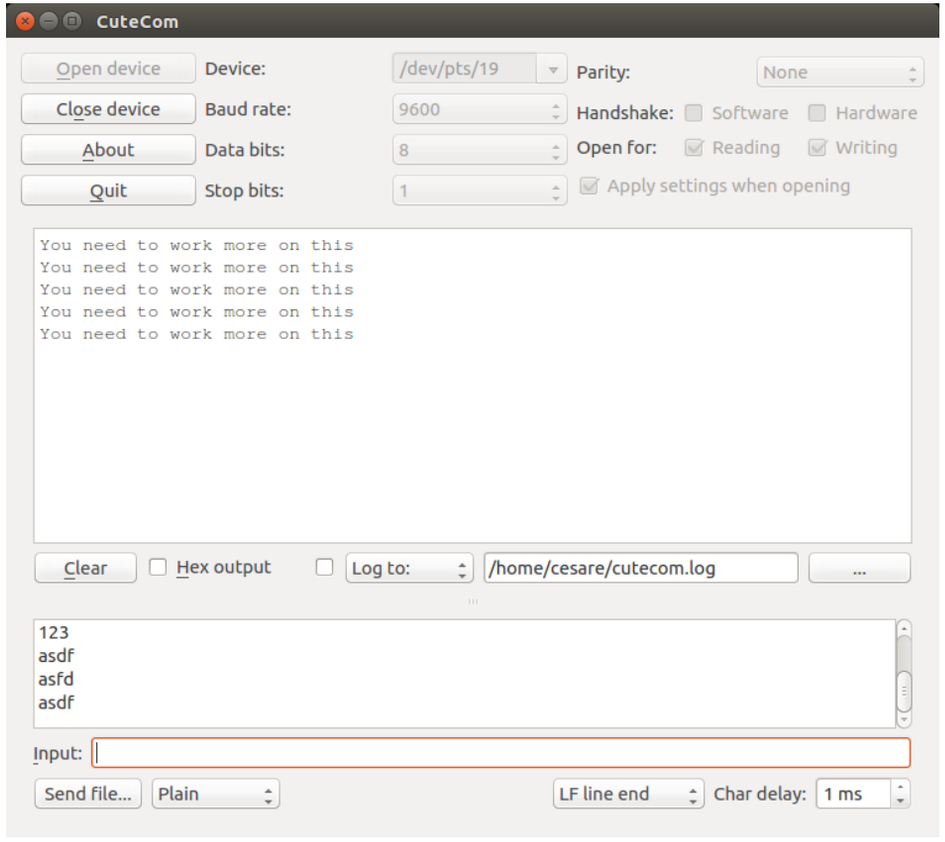
Tools used:

→ simavr
→ avr-gdb

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: level 2 – Debugging





ARDUINO DAY 2019

DEMO

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: level 3 – Disassembler

SORINT.lab

Tools used:

→ avr-objdump
→ radare2

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: level 3 – Assembly Crash Course (2 slides)

- Registers: we have 32 general purpose 8-bit registers (r0-r31). All logic and arithmetic ops operates on these. RAM is accessed with load and store operations
- Special Registers:
  - PC: 16- or 22 bit program counter (holds **next** instruction to be executed)
  - SP: 8- or 16 bit stack pointer
  - SREG: 8 bit status register
- Status bits:
  - 0: C → Carry Flag
  - 1: Z → Zero Flag
  - 2: N → Negative Flag
  - 3: V → Overflow Flag
  - 4: S → Sign Flag
  - 5: H → Half carry flag (BCD arithmetic)
  - 6: T: → T bit copy (BST opcode)
  - 7: I: → Interrupt flag

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: level 3 – Assembly Crash Course (2 slides)

Instruction syntax:

```
mov r10, r0          ; move content of r0 in r10

out 0x0b, r0         ; write the content of r0 in the specified port
```

It uses „Intel Syntax“, with destination before source.

Opcodes are documented in AVR site:

DEC

16-bit Opcode:

1001	010d	dddd	1010

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: level 3 – Disassembler

Interrupt Vector Table Description

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	PCINT0	Pin Change Interrupt Request 0
11	$0014	PCINT1	Pin Change Interrupt Request 1
12	$0016(3)	PCINT2	Pin Change Interrupt Request 2
13	$0018	WDT	Watchdog Time-out Interrupt
14	$001A	TIMER2 COMPA	Timer/Counter2 Compare Match A
15	$001C	TIMER2 COMPB	Timer/Counter2 Compare Match B
16	$001E	TIMER2 OVF	Timer/Counter2 Overflow

Interrupt Vector Table Disassembly

```
00000000 <.sec1>:
   0:   0c 94 35 00     jmp     0x6a    ;  0x6a
   4:   0c 94 5d 00     jmp     0xba    ;  0xba
   8:   0c 94 5d 00     jmp     0xba    ;  0xba
   c:   0c 94 5d 00     jmp     0xba    ;  0xba
  10:   0c 94 5d 00     jmp     0xba    ;  0xba
  14:   0c 94 5d 00     jmp     0xba    ;  0xba
  18:   0c 94 5d 00     jmp     0xba    ;  0xba
  1c:   0c 94 5d 00     jmp     0xba    ;  0xba
  20:   0c 94 5d 00     jmp     0xba    ;  0xba
  24:   0c 94 5d 00     jmp     0xba    ;  0xba
  28:   0c 94 5d 00     jmp     0xba    ;  0xba
  2c:   0c 94 5d 00     jmp     0xba    ;  0xba
  30:   0c 94 5d 00     jmp     0xba    ;  0xba
  34:   0c 94 5d 00     jmp     0xba    ;  0xba
  38:   0c 94 5d 00     jmp     0xba    ;  0xba
  3c:   0c 94 5d 00     jmp     0xba    ;  0xba
  40:   0c 94 26 02     jmp     0x44c   ;  0x44c
  44:   0c 94 5d 00     jmp     0xba    ;  0xba
  48:   0c 94 f4 01     jmp     0x3e8   ;  0x3e8
  4c:   0c 94 ce 01     jmp     0x39c   ;  0x39c
  50:   0c 94 5d 00     jmp     0xba    ;  0xba
```

Arduino RE: level 3 – Disassembler

SORINT lab

Entry Point

```
a4:   21 97           sbiw    r28, 0x01       ; 1
a6:   fe 01           movw    r30, r28
a8:   0e 94 85 03     call    0x70a   ;  0x70a
ac:   c4 33           cpi     r28, 0x34       ; 52
ae:   d1 07           cpc     r29, r17
b0:   c9 f7           brne    .-14            ;  0xa4
b2:   0e 94 70 02     call    0x4e0   ;  0x4e0
b6:   0c 94 90 03     jmp     0x720   ;  0x720
ba:   0c 94 00 00     jmp     0       ;  0x0
be:   cf 92           push    r12
c0:   df 92           push    r13
c2:   ef 92           push    r14
c4:   ff 92           push    r15
```

```
$ avr-objdump -j .sec1 -d -m avr5 –disassemble-zeroes <HEX file name>
```

ARDUINO DAY 2019

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: level 3 – Disassembler

```
|      ::   0x000005ce      b105           cpc r27, r1             ; compare with carry
|      `==< 0x000005d0      98f3           brcs 0x5b8              ; branch if carry flag is set
|      :    0x000005d2      21e0           ldi r18, 0x01           ; LDI Rd,K. load immediate
|      :    0x000005d4      c21a           sub r12, r18            ; subtract without carry
|      :    0x000005d6      d108           sbc r13, r1             ; subtract with carry
|      :    0x000005d8      e108           sbc r14, r1             ; subtract with carry
|      :    0x000005da      f108           sbc r15, r1             ; subtract with carry
|      :    0x000005dc      88ee           ldi r24, 0xe8           ; LDI Rd,K. load immediate
|      :    0x000005de      880e           add r8, r24             ; add without carry
|      :    0x000005e0      83e0           ldi r24, 0x03           ; LDI Rd,K. load immediate
|      :    0x000005e2      981e           adc r9, r24             ; add with carry
|      :    0x000005e4      a11c           adc r10, r1             ; add with carry
|      :    0x000005e6      b11c           adc r11, r1             ; add with carry
|      :    0x000005e8      c114           cp r12, r1              ; compare
|      :    0x000005ea      d104           cpc r13, r1             ; compare with carry
|      :    0x000005ec      e104           cpc r14, r1             ; compare with carry
|      :    0x000005ee      f104           cpc r15, r1             ; compare with carry
|      `=<  0x000005f0      19f7           brne 0x5b8              ; branch if not equal
|           0x000005f2      80e2           ldi r24, 0x20           ; LDI Rd,K. load immediate
|           0x000005f4      91e0           ldi r25, 0x01           ; LDI Rd,K. load immediate
|           0x000005f6      0e949401       call fcn.00000328       ; long call to a subroutine
|           0x000005fa      a0e6           ldi r26, 0x60           ; '`' ; LDI Rd,K. load immediate
|           0x000005fc      b1e0           ldi r27, 0x01           ; LDI Rd,K. load immediate
|           0x000005fe      e3e1           ldi r30, 0x13           ; LDI Rd,K. load immediate
|           0x00000600      f1e0           ldi r31, 0x01           ; LDI Rd,K. load immediate
|           0x00000602      8dea           ldi r24, 0xad           ; LDI Rd,K. load immediate
|           0x00000604      9cee           ldi r25, 0xec           ; LDI Rd,K. load immediate
|           ;-- hit0_0:
|           ; CODE XREF from fcn.000004b4 (0x60e)
|      .-> 0x00000606      8927           eor r24, r25            ; exclusive OR
|      :    0x00000608      8d93           st x+, r24              ; ST X,Rr. store indirect
|      :    0x0000060a      8191           ld r24, z+              ; LD Rd,X. load indirect
|      ,==< 0x0000060c      8111           cpse r24, r1            ; compare, skip if equal
|      |`=< 0x0000060e      fbcf           rjmp 0x606              ; relative jump
|      |    ; CODE XREF from fcn.000004b4 (0x60c)
|      `--> 0x00000610      c0e6           ldi r28, 0x60           ; '`' ; LDI Rd,K. load immediate
|           ; CODE XREF from fcn.000025c8 (0x25fe)
|           0x00000612      d1e0           ldi r29, 0x01           ; LDI Rd,K. load immediate
|           ; CODE XREF from fcn.000004b4 (0x682)
|           0x00000614      83e9           ldi r24, 0x93           ; LDI Rd,K. load immediate
|           0x00000616      91e0           ldi r25, 0x01           ; LDI Rd,K. load immediate
|           0x00000618      0e94e300       call fcn.000001c6       ; long call to a subroutine
[0x0000006a]>
```

MICROCHIP

AVR Assembler Instructions

CONTENTS | SEARCH

- Documentation Home
- AVR Assembler
  - Preface
  - AVR Assembler Known Issues
  - AVR Assembler Command Line Options
  - Assembler source
  - AVR Assembler Syntax
  - Assembler directives
  - AVR Assembler Preprocessor
  - Expressions
  - Instruction mnemonics
  - Instructions
    - ADC - Add with Carry
    - ADD - Add without Carry
    - ADIW - Add Immediate to Word
    - AND - Logical AND
    - ANDI - Logical AND with Immediateand
    - ASR - Arithmetic Shift Right
    - BCLR - Bit Clear in SREG
    - BLD - Bit Load from the T

EOR - Exclusive OR

Description:
Performs the logical EOR between the contents of register Rd and register Rr and places the result in the destination register Rd.

Operation:

Rd ← Rd ⊕ Rr

Syntax: Operands: Program Counter:

EOR Rd,Rr 0 ≤ d ≤ 31, 0 ≤ r ≤ 31 PC ← PC + 1

16-bit Opcode:

0010	01rd	dddd	rrrr

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S: N ⊕ V, For signed tests.

V:0

Cleared

N:R7

Set if MSB of the result is set; cleared otherwise.

ARDUINO DAY 2019

DEMO

March 16TH
day.arduino.cc
#ArduinoD19

Arduino RE: level 3 – Reference

- ATMEL AVR datasheets
- AVR Assembly:
  - https://www.microchip.com/webdoc/avrassembler/avrassembler.wb_instruction_list.html
  - https://en.wikipedia.org/wiki/Atmel_AVR_instruction_set
  - http://www.avr-tutorials.com
- Tools
  - https://radare.org
  - http:s//github.com/buserror/simavr
  - https://blog.didierstevens.com/programs/xorsearch/

- This presentation:
  - https://github.com/cecio/

March 16TH
day.arduino.cc
#ArduinoD19

Cesare Pizzi

Thank you!

March 16TH
day.arduino.cc
#ArduinoD19