INTRODUCTION TO AVR

Gholamhossein Tavasoli

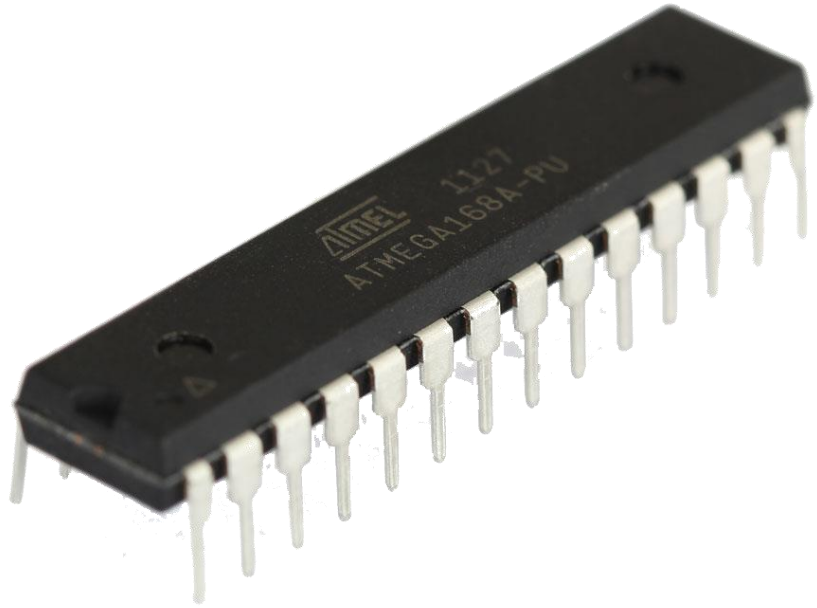
@University of Zanjan

HISTORY

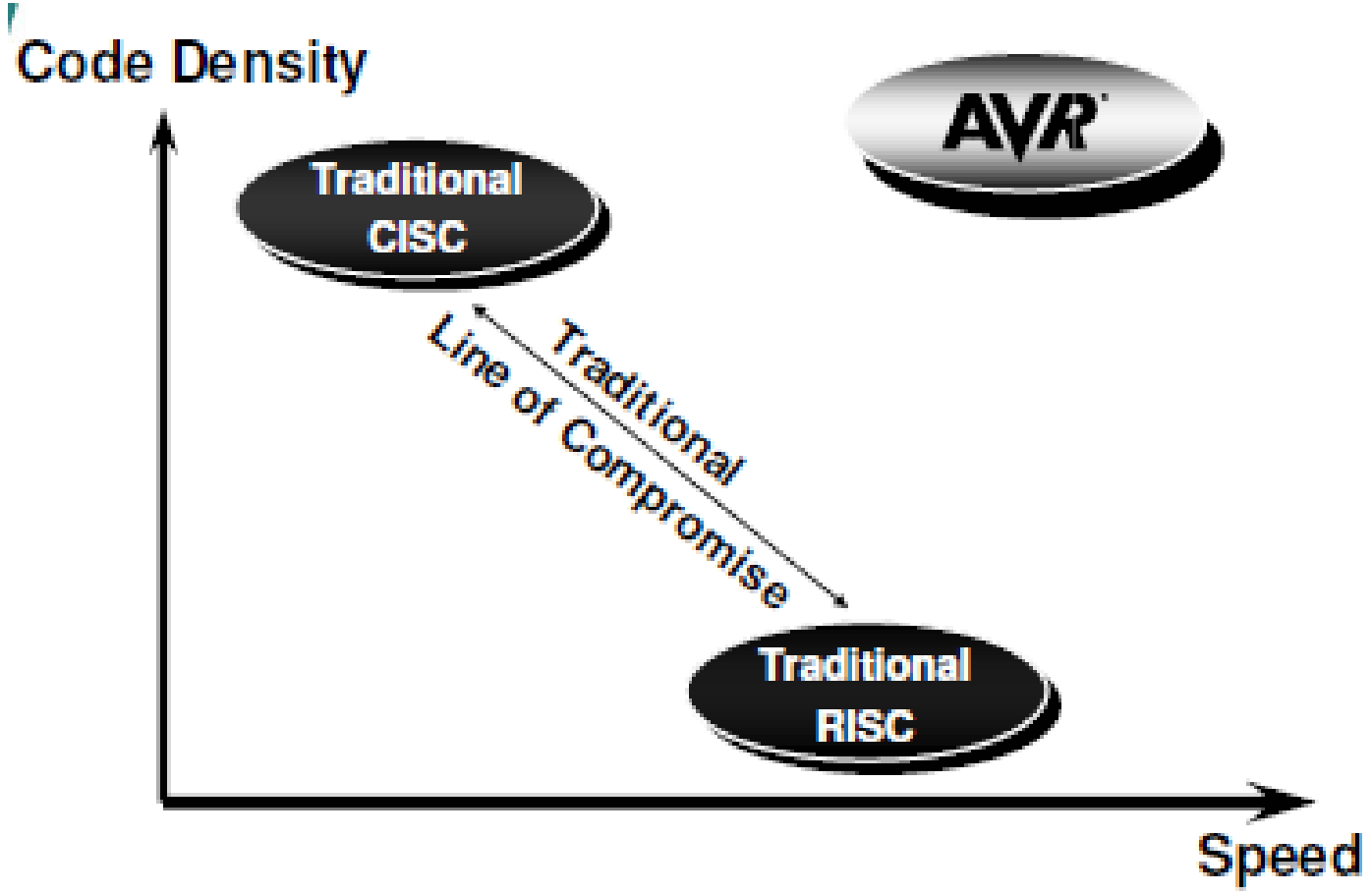
- Atmel AVR is a family of 8-bit RISC microcontrollers produced by Atmel in **1996**
- The AVR architecture was conceived by two students at the **Norwegian Institute of Technology (NTH)** and further refined and developed at **Atmel Norway**, the Atmel daughter company founded by the two chip architects.
- Atmel says that the name AVR is **not** an acronym and does not stand for anything in particular.
- However, it is commonly accepted that AVR stands for **A**lf (Egil Bogen) and **V**egard (Wollan)'s **R**ISC processor.

AVR KEY FEATURES

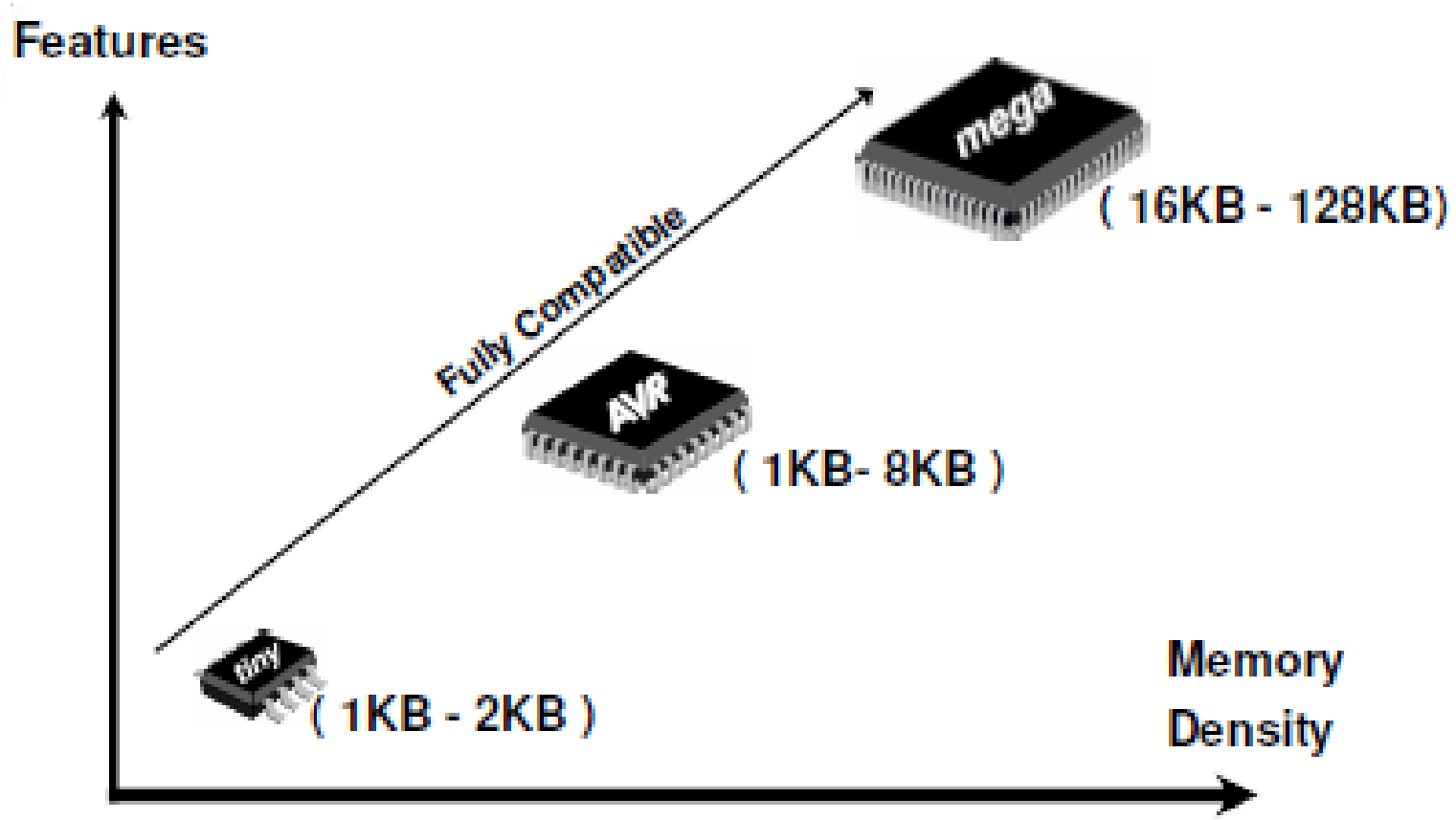
- High Performance 8 bit MCU
- RISC Architecture
  - 32 Registers
  - Single Cycle Execution
- Low Power (CMOS)
- Efficient C Language Code Density
- On-chip in-system programmable
- RISC Performance with CISC Code Density (program memory usage)



BREAKING TRADITIONS



THE THREE AVR FAMILIES



ATTINY AVR PRODUCTS

	tiny11	tiny12	tiny15	tiny28
Pins	8	8	8	28/32
Flash	1 KB	1 KB	1 KB	2 KB
EEPROM	-	64 B	64 B	-
PWMs	-	-	1	1
ADC	-	-	4@ 10-bit	-

CLASSIC AVR PRODUCTS

	S1200	S2323	S2343	S2313
Pins	20	8	8	20
Flash	1 KB	2 KB	2 KB	2 KB
SRAM	-	128 B	128 B	128 B
EEPROM	64 B	128 B	128 B	128 B
PWMs	-	-	-	1
UART	-	-	-	1

CLASSIC AVR PRODUCTS

	S4433	S8515	VC8534	S8535
Pins	28/32	40/44	48	40/44
Flash	4 KB	8 KB	8 KB	8 KB
SRAM	128 B	512 B	256 B	512 B
EEPROM	256 B	512 B	512 B	512 B
PWMs	1	2	-	2
UART	1	1	-	1
ADC	6@ 10-bit	-	6@ 10-bit	8@ 10-bit

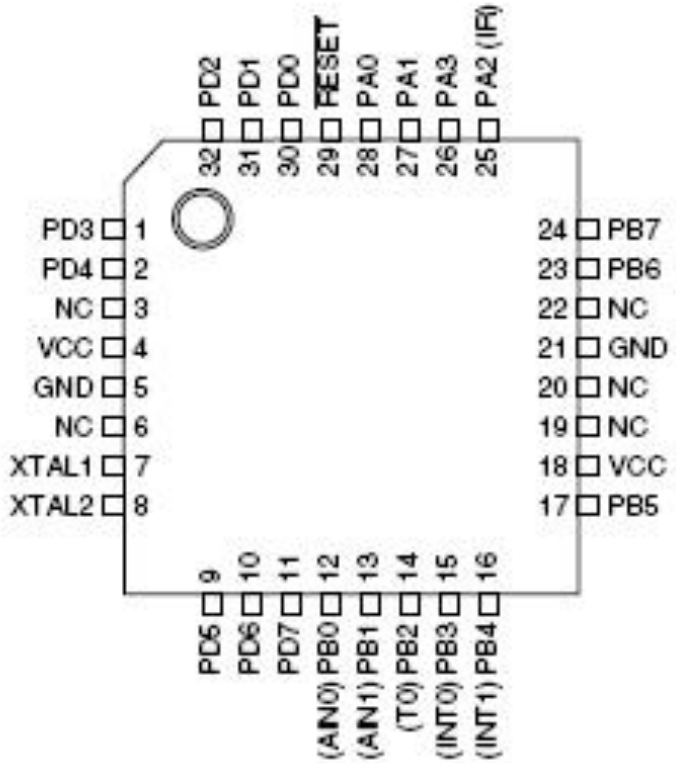
MEGA AVR PRODUCTS

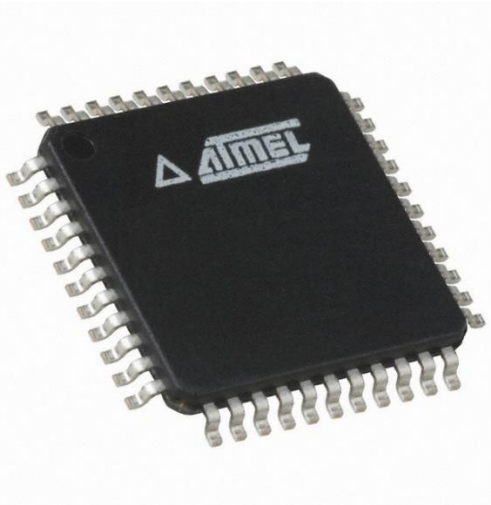
	mega 8	mega 16	mega 32	mega 64	mega 128
Pins	28/32	40/44	40/44	64	64
Flash	8 KB	16 KB	32 KB	64 KB	128 KB
SRAM	1 KB	1 KB	2 KB	4 KB	4 KB
EEPROM	512 B	512 B	1 KB	2 KB	4KB
TWI	1	1	1	1	1
U(S)ART	1	1	1	2	2
ADC	8@ 10-bit	8@ 10-bit	8@ 10-bit	8@ 10-bit	8@ 10-bit
PWM	3	4	4	8	8
JTAG	-	Yes	Yes	Yes	Yes
Self Program	Yes	Yes	Yes	Yes	Yes

PACKAGES

PDIP

TQFP/QFN/MLF





WHICH ONE?

- Application (Define Requirements)
- Frequency and Voltage Range
  - Very Low Power (1.8-5.5V)
  - Low Power (2.7-5.5V)
- Memory
- Packages

ATTINY 12 (FEATURES)

- 8 pin package
- 1 KBytes ISP Flash
- 64 Bytes ISP EEPROM
- Up to 6 programmable I/O lines
- 8 Bit Timer/Counter
- 1 External Interrupt
- Interrupt and wake up on any pin change
- Analog Comparator
- Internal Accurate Oscillator with calibratable frequency

ATTINY 12 (TYPICAL APPLICATIONS)

- Secure EEPROM
- Keyless Entry for Car Alarm
- Fire Detector
- Toys
- DIP switch replacement
- Protocol Converter
- Motor Control
- Replacing External Logic
- Signal Processing Applications
- Data Logger
- Coprocessor

ATTINY 28 (FEATURES)

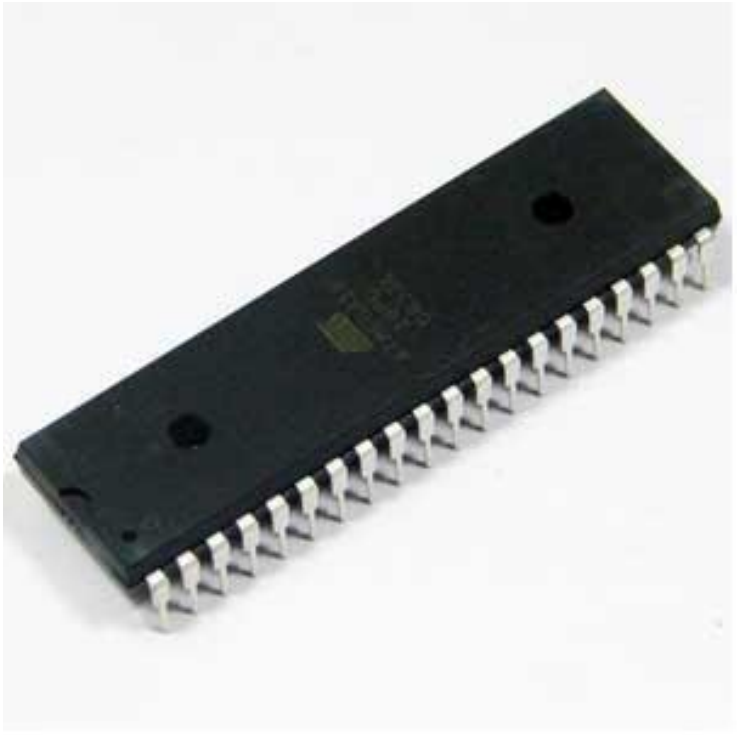
- 28/32 pin packages
- 2 KBytes Flash
- 11 programmable I/O lines, 8 dedicated input lines and 1 dedicated output line
- 8 Bit Timer/Counter
- 2 External Interrupts
- Interrupt and wake up on any pin change
- Built-in High-current LED Driver with Programmable Modulation
- Analog Comparator
- Internal Accurate Oscillator with calibratable frequency
- Operates down to 1.8V

ATTINY 28 (TYPICAL APPLICATIONS)

- Remote Control
- Keyboard Controller
- I/O Controller
- Communication Equipment
- Low-cost/High pin count Applications

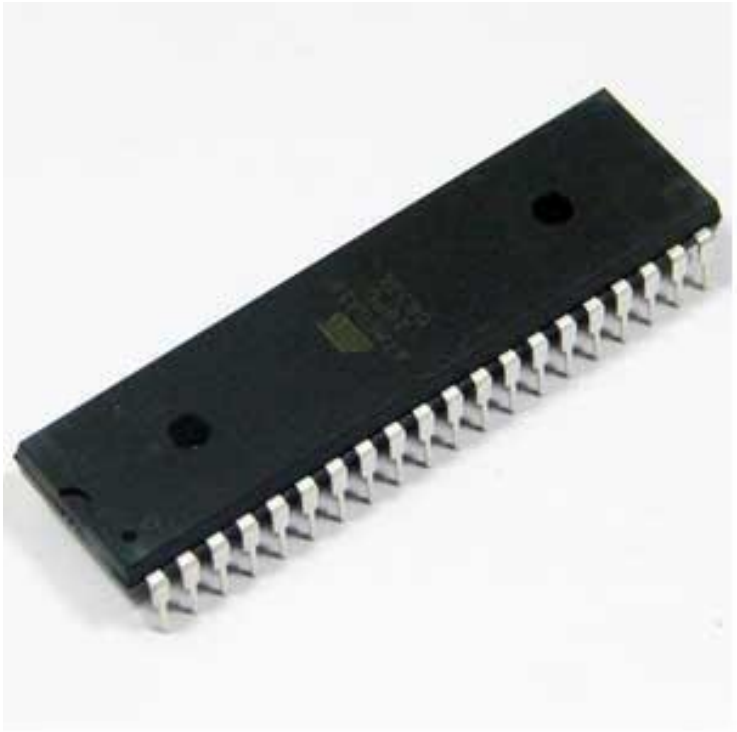
ATMEGA 16 (FEATURES)

- 40/44 pin packages
- 16 KBytes ISP Flash, Self Programmable
- 512 Bytes ISP EEPROM
- 1 KBytes SRAM
- Full Duplex UART
- SPI – Serial Interface
- TWI – Serial Interface
- 8- and 16-bits Timer/Counters with PWM
- 2 External Interrupts
- 10-bit ADC with 8 Multiplexed Inputs
- RTC with Separate 32 kHz Oscillator
- Analog Comparator
- JTAG Interface with On-Chip Debugger



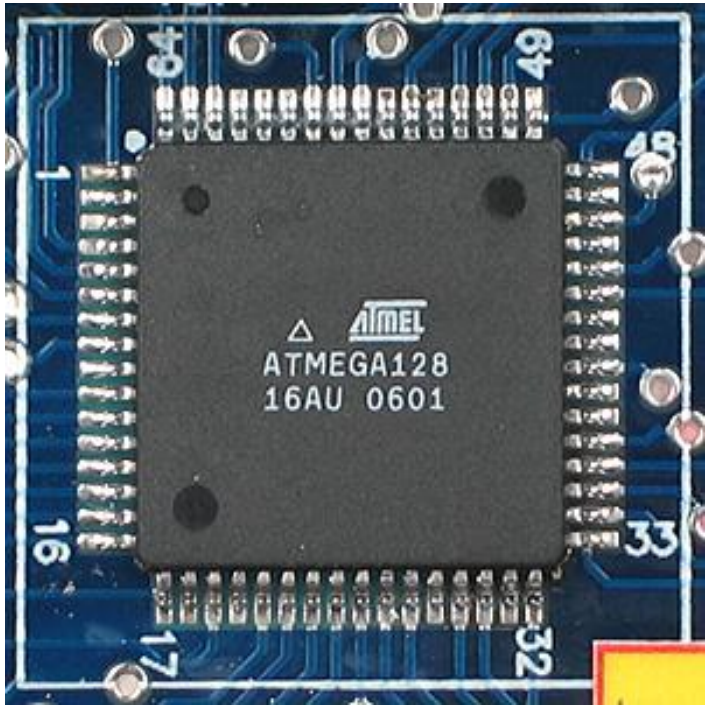
ATMEGA 16 (TYPICAL APPLICATIONS)

- Smart Battery
- Advanced Battery Charger
- Power Meter
- Temperature Logger
- Voltage Logger
- Tension Control
- Touch Screen Sensor
- Metering Applications
- UPS
- 3 Phase Motor Controller
- Industrial Control
- Power Management



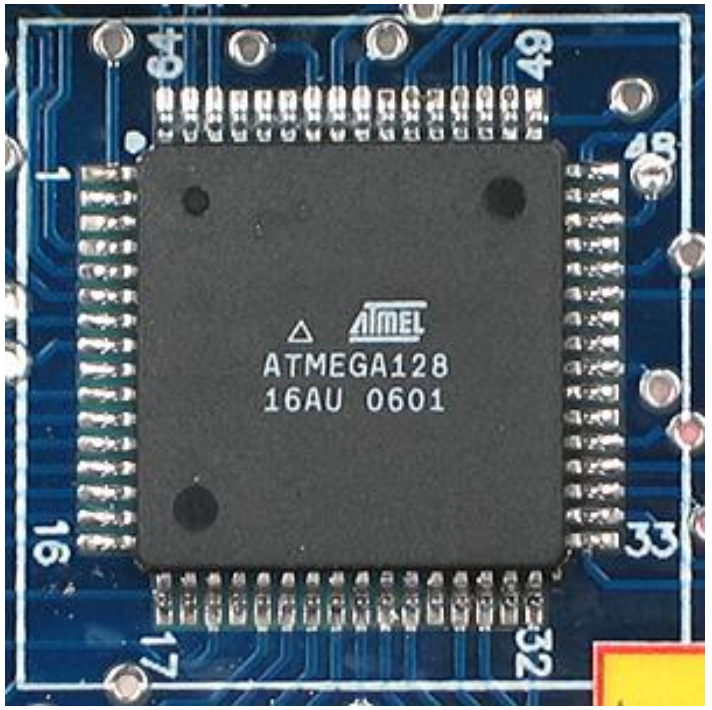
ATMEGA 128 (FEATURES)

- 64 pin package (48 I/O, 16 Special Function)
- 128 KBytes ISP Flash , Self Programmable
- 4 KBytes SRAM
- 4 KBytes ISP EEPROM
- Dual Full Duplex USARTs
- SPI – Serial Interface
- TWI – Serial Interface
- 8- and 16-bits Timer/Counters with PWM
- 2 External Interrupts
- 10-bit ADC with 8 Multiplexed Inputs
- RTC with Separate 32 kHz Oscillator
- Analog Comparator
- JTAG Interface with On-Chip Debugger



ATMEGA 128 (TYPICAL APPLICATIONS)

- Analog Telephone
- Sensor Applications
- Automobile Applications
- Telecom Applications
- 3-phase Motor Control
- GPS
- Industrial Control



A C CODE EXAMPLE

- The following example illustrates how the AVR benefits in terms of:
  - Code Size
  - Throughput
  - Power Consumption

A SMALL C FUNCTION

```
/* Return the maximum value of a table of 16 integers */

int max(int *array)
{
  char a;
  int maximum=-32768;

  for (a=0;a<16;a++)
     if (array[a]>maximum)
        maximum=array[a];
  return (maximum);
}
```

AVR ASSEMBLY OUTPUT

- Code Size: **46 Bytes**, Execution time: **335 cycles**

```
; 7.      for (a=0;a<16;a++)
         LDI     R18,LOW(0)
         LDI     R19,128
         CLR     R22
?0001:
         CPI     R22,LOW(16)
         BRCC    ?0000
; 8.       {
; 9.         if (array[a]>maximum)
         MOV     R30,R22
         CLR     R31
         LSL     R30
         ROL     R31
         ADD     R30,R16
         ADC     R31,R17
         LDD     R20,Z+0
         LDD     R21,Z+1
         CP      R18,R20
         CPC     R19,R21
         BRGE    ?0005
; 10.           maximum=array[a];
         MOV     R18,R20
         MOV     R19,R21
?0005:
         INC     R22
         RJMP    ?0001
?0000:
; 11.      }
; 12.      return (maximum);
         MOV     R16,R18
         MOV     R17,R19
; 13.    }
         RET
```

C51 ASSEMBLY OUTPUT

- Code Size: **112 Bytes**, Execution time: **9384 cycles**

```
; FUNCTION _max (BEGIN)
;---- Variable 'array' assigned to Register 'R1/R2/R3' ----
     ; SOURCE LINE # 4
     ; SOURCE LINE # 5
     ; SOURCE LINE # 7
     MOV     maximum,#080H
     MOV     maximum+01H,#00H
     ; SOURCE LINE # 9
;---- Variable 'a' assigned to Register 'R5' ----
     CLR     A
     MOV     R5,A
?C0001:
     ; SOURCE LINE # 10
     ; SOURCE LINE # 11
     MOV     A,R5
     MOV     R7,A
     RLC     A
     SUBB    A,ACC
     MOV     R6,A
     MOV     A,R7
     ADD     A,ACC
     MOV     R7,A
     MOV     A,R6
     RLC     A
     MOV     DPL,R7
     MOV     DPH,A
     LCALL   ?C?ILDOPTR
     MOV     R7,A
     MOV     R6,B
     SETB    C
     SUBB    A,maximum+01H
     MOV     A,maximum
     XRL     A,#080H
     MOV     R0,A
     MOV     A,R6
     XRL     A,#080H
     SUBB    A,R0
     JC      ?C0003
     ; SOURCE LINE # 12
     MOV     maximum,R6
     MOV     maximum+01H,R7
                                        ; SOURCE LINE # 13
?C0003:
     INC     R5
     CJNE    R5,#010H,?C0001
?C0002:
                                        ; SOURCE LINE # 14
     MOV     R6,maximum
MOV     R7,maximum+01H
                                        ; SOURCE LINE # 15
?C0005:
     RET
    ; FUNCTION _max (END)
?C?ILDOPTR:
    CJNE     R3,#0x01,0x195
    MOV      A,0x82
    ADD      A,R1
    MOV      0x82,A
    MOV      A,0x83
    ADDC     A,R2
    MOV      0x83,A
    MOVX     A,@DPTR
    MOV      0xF0,A
    INC        DPTR
    MOVX     A,@DPTR
    RET
    JNC      0x1A0
    MOV      A,R1
    ADD      A,0x82
    MOV      R0,A
    MOV      0xF0,@R0
    INC      R0
    MOV      A,@R0
    RET
    CJNE     R3,#0xFE,0xAD
    MOV      A,R1
    ADD      A,0X82
    MOV      R0,A
    MOV      A,@R0
    MOV      0xF0,A
    INC      R0
    RET
    MOV      A,0X83
    ADD      A,R2
    MOV      0X83,A
    MOV      A,R1
    MOVC     A,@A+DPTR
    RET
```

HC11 ASSEMBLY OUTPUT

- Code Size: **57 Bytes**, Execution time: **5244 cycles**


```
; 7.      for (a=0;a<16;a++)
        TSX
        LDD     #-32768
        STD     1,X
        CLR     0,X
?0001:
        LDAA    0,X
        CMPA    #16
        BHS     ?0000
; 8.      {
; 9.        if (array[a]>maximum)
        PSHY
        TAB
        CLRA
        LSLD
        TSX
        ADDD    0,X
        XGDX
        LDD     0,X
        TSX
        STD     5,X
        INS
        INS
        CPD     3,X
        BLE     ?0005
; 10.         maximum=array[a];
        STD     3,X
?0005:
        TSX
        INC     0,X
        BRA     ?0001
?0000:
; 11.     }
; 12.     return (maximum);
        LDD     1,X
; 13.   }
        PULX
        PULX
        INS
        PULY
        RTS
```

PIC16C74 ASSEMBLY OUTPUT

- Code Size: **87 Bytes**, Execution time: **2492 cycles**

```
        bcf     3,5
        movwf   ?a_maxnum& (0+127)
;MAX_MAIN.C: 4: char a;
;MAX_MAIN.C: 5: int maximum=-32768;
        clrf    (?a_maxnum+2)& (0+127)
        movlw   128
        movwf   (?a_maxnum+3)& (0+127)
;MAX_MAIN.C: 7: for (a=0;a<16;a++)
        clrf    (?a_maxnum+1)& (0+127)
12
;MAX_MAIN.C: 8: {
;MAX_MAIN.C: 9: if (array[a]>maximum)
        bcf     3,5
        movf    (?a_maxnum+1)& (0+127),w
        addwf   (?a_maxnum+1)& (0+127),w
        addwf   ?a_maxnum& (0+127),w
        movwf   4
        movf    0,w
        movwf   btemp
        incf    4
        movf    0,w
        movwf   btemp+1
        movf    (?a_maxnum+3)& (0+127),w
        xorlw   128
        movwf   btemp+2
        movf    btemp+1,w
        xorlw   128
        subwf   btemp+2,w
        btfss   3,2
        goto    u15
        movf    btemp,w
        subwf   (?a_maxnum+2)& (0+127),w
u15
        btfsc   3,0
        goto    15
;MAX_MAIN.C: 10: maximum=array[a];
        bcf     3,5
        movf    (?a_maxnum+1)& (0+127),w
        addwf   (?a_maxnum+1)& (0+127),w
        addwf   ?a_maxnum& (0+127),w
        movwf   4
        movf    0,w
        movwf   (?a_maxnum+2)& (0+127)
        incf    4
        movf    0,w
        movwf   (?a_maxnum+3)& (0+127)
15
;MAX_MAIN.C: 11: }
        bcf     3,5
        incf    (?a_maxnum+1)& (0+127)
        movlw   16
        subwf   (?a_maxnum+1)& (0+127),w
        btfss   3,0
        goto    12
;MAX_MAIN.C: 12: return (maximum);
        bcf     3,5
        movf    (?a_maxnum+3)& (0+127),w
        movwf   btemp+1
        movf    (?a_maxnum+2)& (0+127),w
        movwf   btemp
        return
```

PERFORMANCE COMPARISON

- AT90S8515 @ 8 MHz
- 80C51 @ 24 MHz
- 68HC11A8 @ 12 MHz
- PIC16C74 @ 20 MHz

	Code Size (Byte)	Function Exec Time (us)	Current Consumption (mA)
AT90S8515	46 (1)	42 (1)	11 (1)
80C51	112 (2.4)	391 (9)	16 (1.5)
68HC11A8	57 (1.2)	437 (10)	27 (2.5)
PIC16C74	87 (1.9)	125 (3)	13.5 (1.2)

SOME CONCLUSIONS ON THIS CASE

- The C51 would have to run at 224 MHz to match the 8 MHz AVR.
- The HC11 is quite code efficient, but delivers only one 16th of the processing power at more than twice the current consumption
- The PIC is a fast microcontroller, but the AVR delivers more than 3.5 times higher throughput per mW.

WHAT MADE THE AVR DO BETTER?

- Excellent support for 16-bit arithmetic operations
- A lot of registers which eliminate move to and from SRAM (register to register operations)
- Single Cycle execution

INSTRUCTION SET

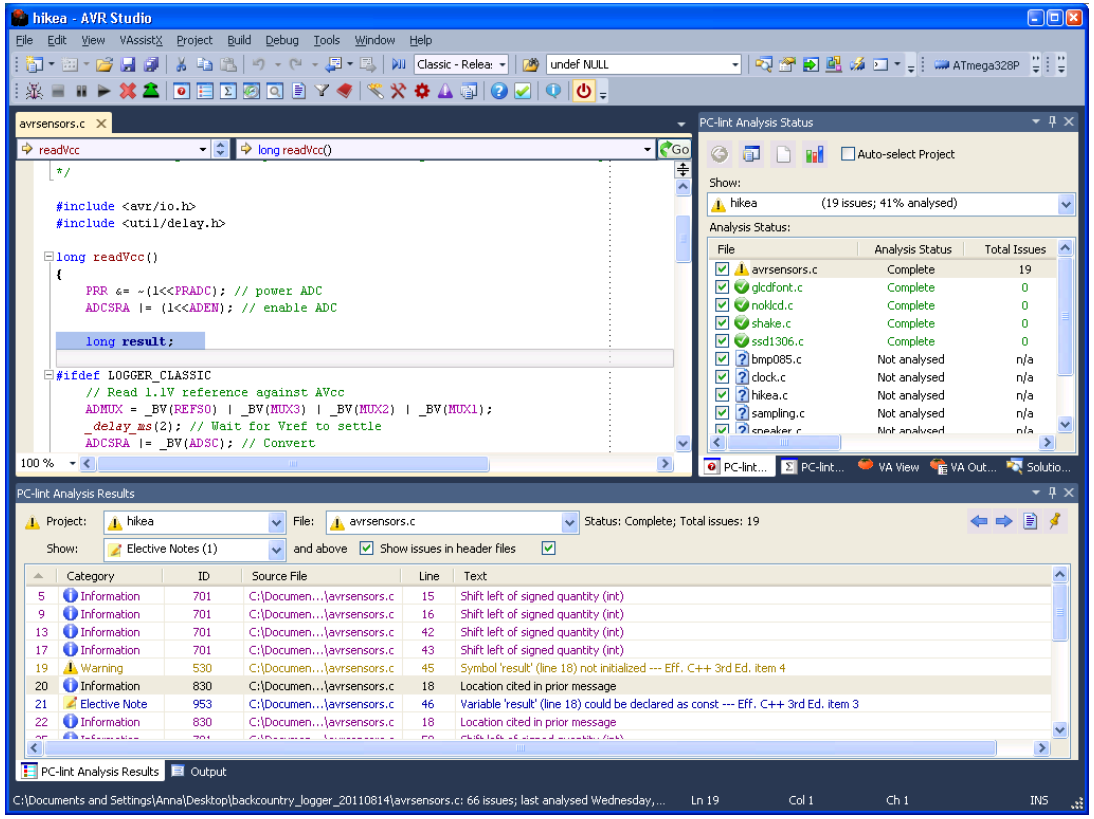
- Instruction Classes
  - Arithmetic/Logic Instructions
  - Data Transfer Instructions
  - Program Control Instructions
  - Bit Set/Test Instructions
- Most Instructions are 16 Bits Wide
- Most Instructions are Executed in One Clock Cycle

DEVELOPMENT TOOLS

- Complete suite of development tools available
- ANSI compliant C Compilers
- Macro-Assemblers
- Linkers/Librarians
- Debuggers/Simulators
- Programmers

AVR STUDIO

- Integrated Development Environment for AVR
- Front end for the AVR Simulator and Emulators
- C and Assembly source level debugging
- Supports third party compilers
- Maintains Project information
- Freely available from www.atmel.com

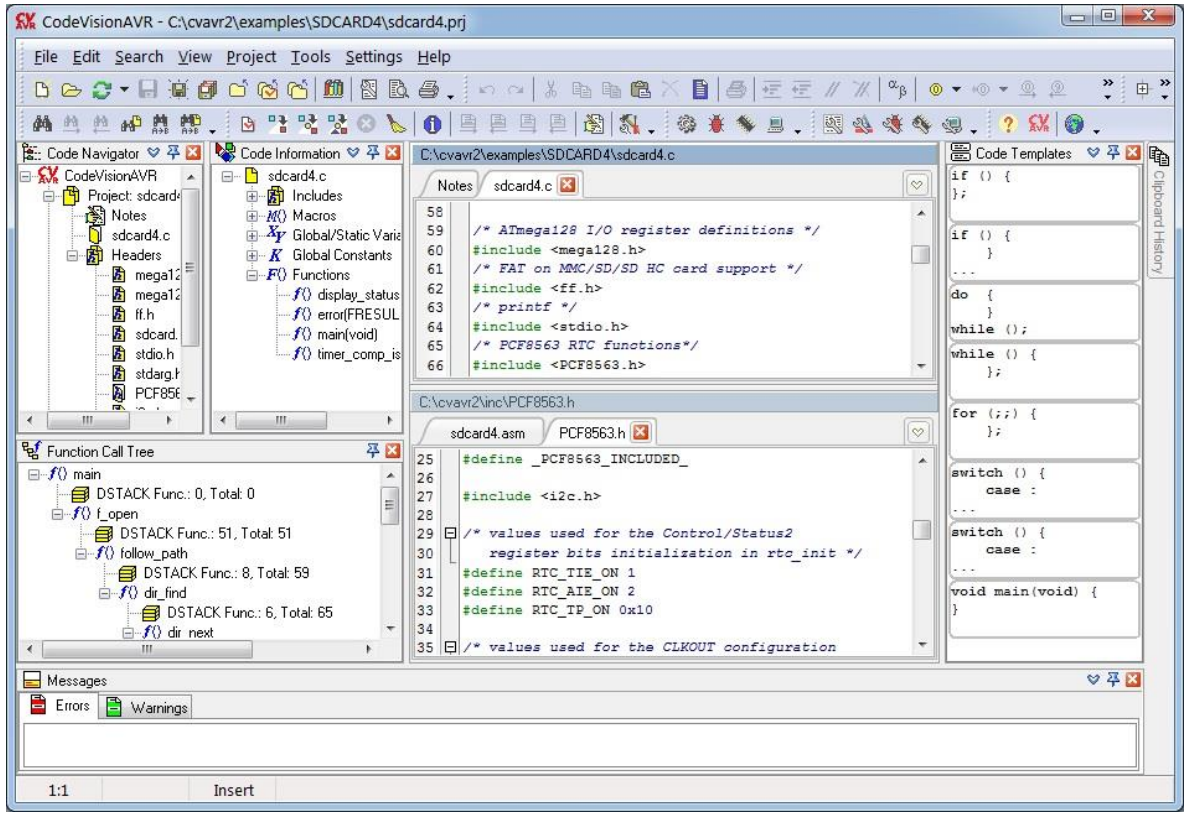


GNU AVR-GCC

- The GNU Ansi C compiler for AVR
- Freeware C compiler (and assembler)
- Supports all AVRs, including tiny devices
- AVR Studio text editor supports integration of AVR-GCC
  - Compile and run the code from AVR Studio
- www.avrfreaks.net/AVRGCC

CODEVISION AVR

- Commercial C Compiler & IDE
- Support JTAG & Parallel (STK200) Programmer
- Code Generator (CodeWizard)
- In System Programming
- Libraries for
  - LCD
  - SPI
  - I2C
  - Sensors
  - EEPROM

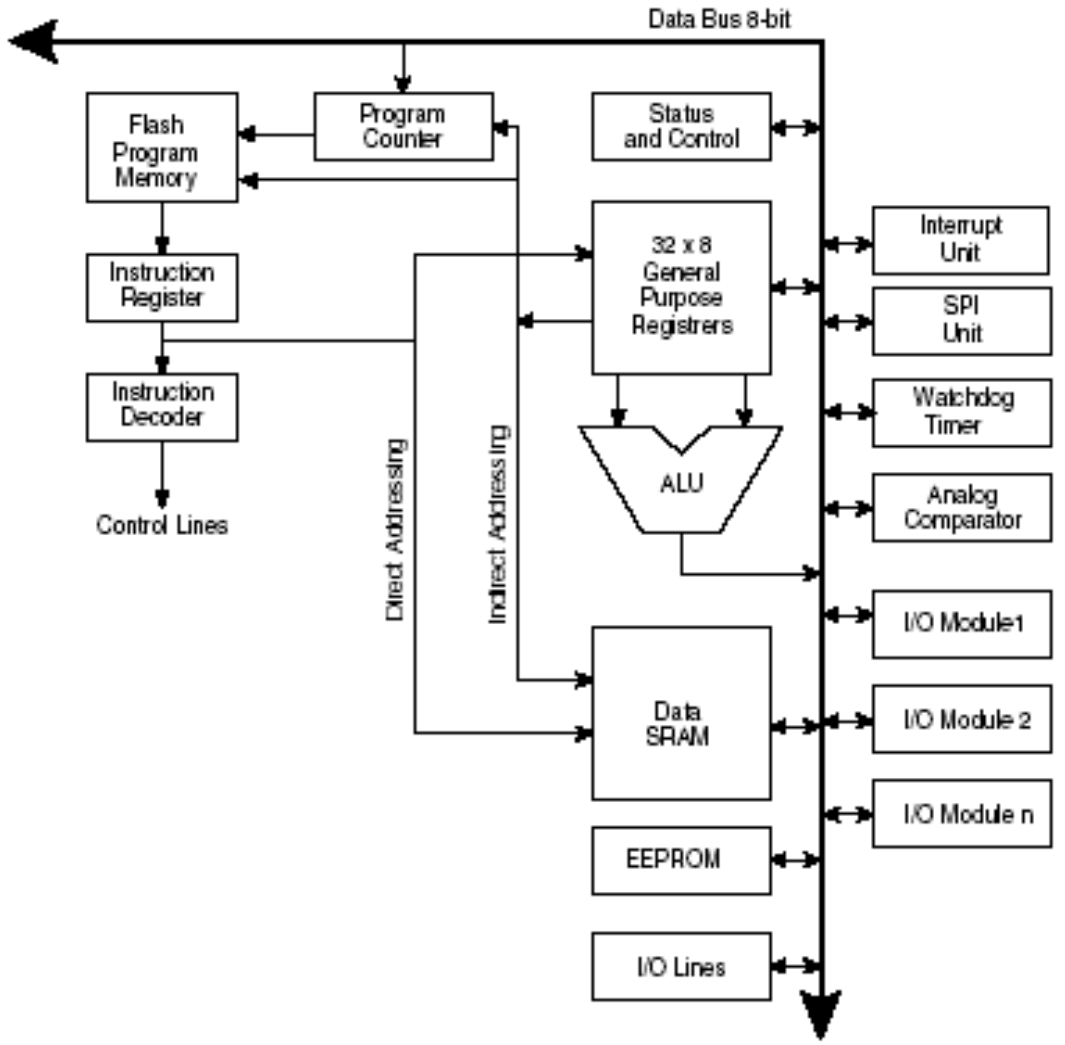


JTAG ICE

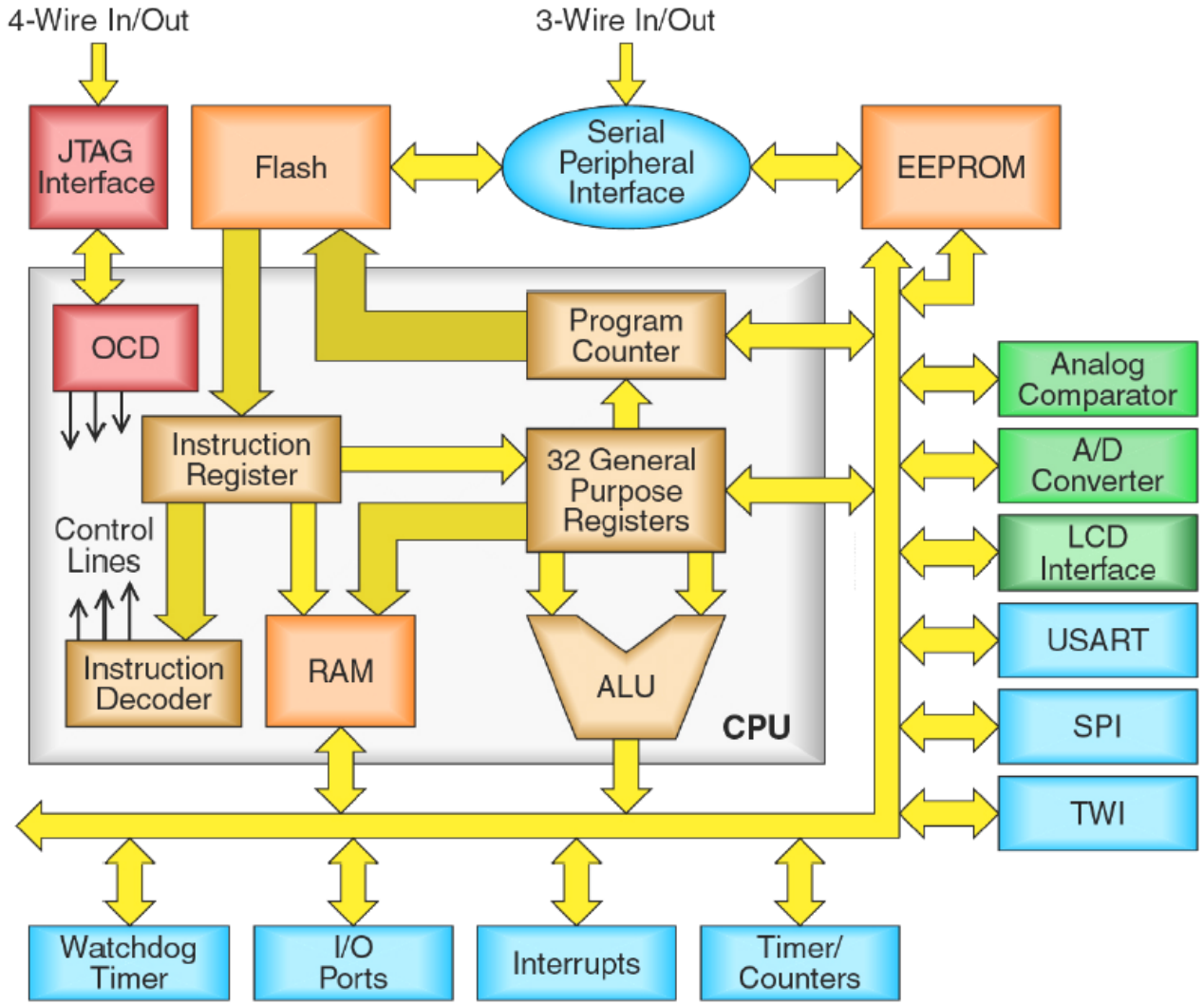
- Controlled by AVR Studio
- Real-Time Emulation in actual silicon
  - Debug the real device at the target board
  - Talks directly to the device through a 4-pin JTAG interface
- Supports
  - Program and Data breakpoints
  - Full execution control
  - Full I/O-view and Watches

AVR RISC ARCHITECTURE

- Single Cycle Instructions: 8mhz = 8mips
- Large register file (32 GP register)
- Every register an accumulator (Output of ALU)
- 3 index register pairs
- Register & IO are mapped in SRAM space
- On Level Pipelining (Execution + Prefetch)



AVR BLOCK DIAGRAM



HARVARD ARCHITECTURE

- The AVR has Harvard architecture where program and data are stored in separate physical memory systems that appear in different address spaces. (Compare with Von Neumann Architecture)
- The AVR having the ability to read data items from program memory using special instructions.

FIVE MEMORY AREA

- 32 General Purpose 8 bit Registers (32 Bytes)
- Flash Program Memory < 8 MB
  - Vectors, Code, and (Unchangeable) Constant Data
- SRAM Data Memory < 16 MB
  - Runtime Variables and Data
  - Stack Space
- I/O Memory < 64 (224) Bytes
  - Part of SRAM
- EEPROM Data Memory < 16 MB
  - For non-volatile but alterable data