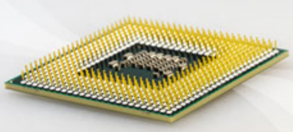
An Overview on ATMEGA32 and AVR Programming

Dr. F. Mohanna
Ali Abbasi

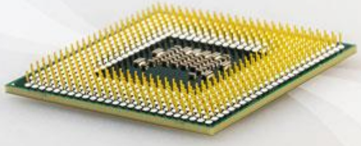
ECE, University of Sistan & Baluchestan

Outline

- ❑ Introduction to Microprocessors and Microcontrollers
- ❑ Introduction to Atmel AVR Family Microcontrollers
- ❑ Atmel AVR ATMega32 Architecture and Organization
- ❑ Starting with a Microcontroller
- ❑ Programming ATMega32 using CodeVisionAVR
- ❑ Programming ATMega328P using Arduino IDE



What is a Microprocessor



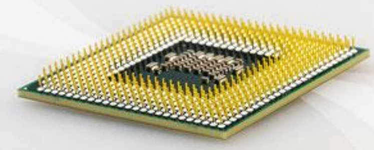
- An integrated circuit that contains all the functions of a central processing unit of a computer.
- Accepts Binary data as input, processes it according to instructions stored in its memory and provides results as output.
- It is generally Multipurpose, Register-based, Clock-driven
- They generally have Von Neumann Architecture.
- They have their specific assembly language for programming.
- **Examples:** Intel 4004, Intel 8086, Intel Core Series, Intel Xeon Series, intel Pentium Series, MIPS R2000, NVIDIA TEGRA Family, ZILOG Z8000, Motorola 68000

An example of Intel 8086 program

```
PAGE      110,100
TITLE     'AVG.asm'
SSEG      SEGMENT     STACK                  'STACK'
DW        32H         DUO(O)
SSEG      ENDS
DSEG      SEGMENT     'DATA'
ORG       92H
DTABLE    DB          62H, 24H, 86H, 24H, 04H, 31H, 74H, 64H, 30H, 99H
SUM       DW          ?
DSEG      ENDS
CSEG      SEGMENT     'CODE'
ASSUM                 SS:SSEG, DS:DSEG, CS:CSEG
MAIN      PROC        FAR
MOV       AX, DSEG
MOV       DS, AX
MOV       CL, 10
MOV       AX, 00H
MOV       DI, OFFSET  DTABLE
LP:       ADD         AX, [DI]
          INC         DI
          DEC         CL
          JNZ         LP
MOV       SUM, AX
MOV       AX, 4C00H
INT       21H
MAIN      ENDP
CSEG      ENDS
END       MAIN
```

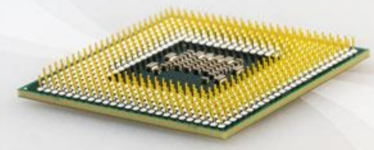
A computer with a microprocessor:

What is System on a Chip (SoC)



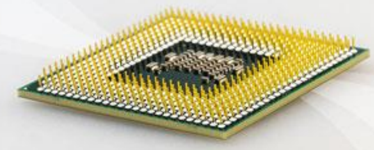
- An integrated circuit (also known as a "chip") that integrates all components of a computer or other electronic system such as central processing unit (CPU), memory, input/output ports.
- Commonly used in embedded systems and the Internet of Things.
- **Examples:** Raspberry Pi, Orange Pi, Beaglebone, Nano Pi, Friendly ARM

What is a Microcontroller



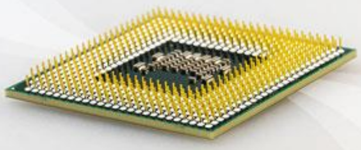
- It is similar to, but less sophisticated than, a system on a chip (SoC).
- They are designed for embedded applications.
- Options range from the simple 4-bit, 8-bit or 16-bit processors to more complex 32-bit or 64-bit processors.
- They generally have Harvard Architecture.
- When they first became available, microcontrollers solely used assembly language. Today, the C programming language is a popular option.
- **Examples:** Intel 8051, Atmel AVR

Microprocessors vs Microcontrollers



- Microcontrollers are designed for embedded applications, While the microprocessors are used for designing general purpose digital computer systems.
- Microprocessors are commonly used as CPU in microcomputer system, whereas microcontrollers are used in minimum component design performing control-oriented applica tions.
- Microprocessors instruction sets are mainly intended to provide for large amounts of data, microcontrollers sets are intended to control input and output.
- Microprocessors design is complex and expensive, microcontrollers design is simple and cost effective.
- Microprocessors design consume more power compared to microcontrollers design.
- Microprocessors instruction set is complex with large number of instructions, whereas microcontrollers has less no of instructions.
- Rapid movement of data between external memory and microprocessors, In microcontrollers movement of data and code within it.
- Program is stored on ROM in Microcontrollers while Microprocessors fetch the code from a secondary storage and load it to RAM.

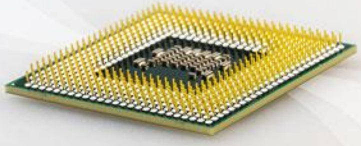
Final Decision



A microcontroller is used where

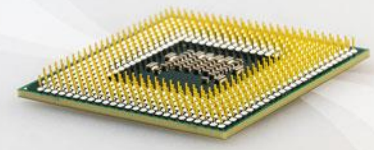
there is a definite input-output relationship.

TOP10 Microcontroller Manufacturers



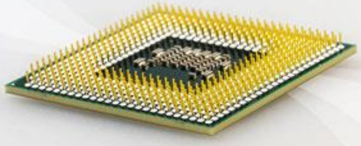
1. Texas Instruments
2. Microchip Company
3. Silicon Labs
4. Renesas Technology Corp
5. Intel Corporation
6. Dallas Semiconductor
7. Fujitsu Semiconductor Europe
8. STMicroelectronics
9. ZiLog Company
10. Freescale Semiconductor Company

Introduction to Atmel AVR



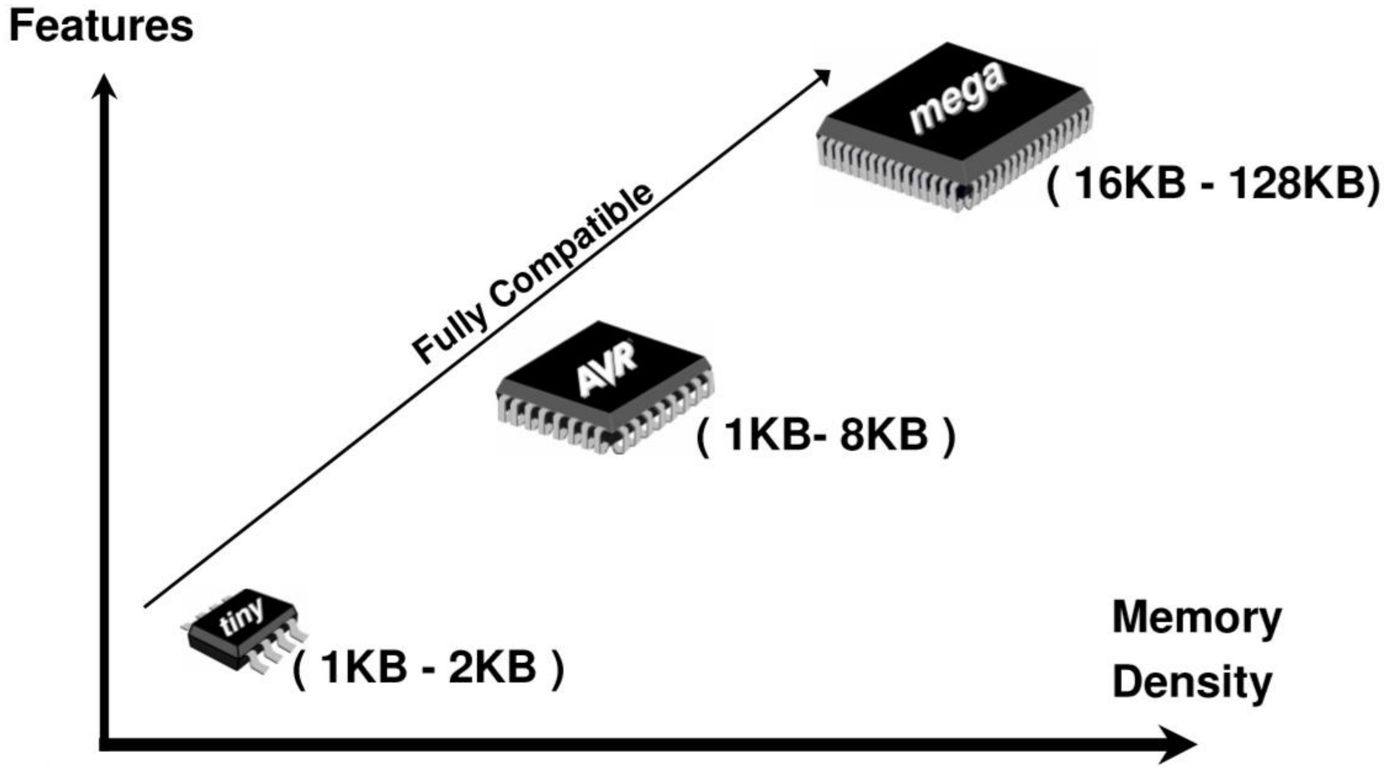
- Atmel Corporation is a manufacturer of semiconductors, founded in 1984.
- Atmel introduced the first 8-bit flash microcontroller in 1993, base on the 8051 core.
- The AVR architecture was conceived by two students at the Norwegian institute of Technology (NTH) Alf-Egil Bogen and Vegard Wollan.
- In 1996, a design office was started in Trondheim, Norway, to work on the AVR series of products.
- Its products include microcontrollers (including 8051 derivatives and AT91SAM and AT91CAP ARM-based micros),and its own Atmel AVR and AVR32 architectures

Introduction to Atmel AVR



- The AVR is a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to one-time programmable ROM, EPROM, EEPROM used by other microcontrollers at the time.
- The AVR is a modified Harvard architecture machine where program and data is stored in separate physical memory systems that appear in different address spaces, but having the ability to read data items from program memory using special instructions.
- Atmel says that the name AVR is not an acronym and does not stand for anything in particular. The creators of the AVR give no definitive answer as to what the term "AVR" stands for, However, it is commonly accepted that **AVR** stands for "**A**lf (Egil Bogen) and **V**egard (Wollan)'s **R**ISC processor".

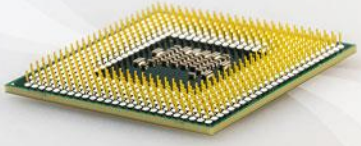
AVR Families



AVR Families: Tiny

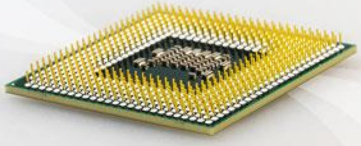
	tiny11	tiny12	tiny15	tiny28
Pins	8	8	8	28/32
Flash	1 KB	1 KB	1 KB	2 KB
EEPROM	-	64 B	64 B	-
PWMs	-	-	1	1
ADC	-	-	4@10-bit	-
Samples	Now	Now	Now	Now
Production	Now	Now	Now	Now

AVR Families: AVR



	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
UART	-	-	-	1
PWM	-	-	-	1
Samples	Now	Now	Now	Now
Production	Now	Now	Now	Now

AVR Families: AVR



	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
UART	1	1	-	1
PWM	1	2	-	2
ADC	6@10-bit	-	6@10-bit	8@10-bit
RTC	-	-	-	Yes
Samples	Now	Now	Now	Now
Production	Now	Now	Now	Now

AVR Families: Mega

	mega161	mega163	mega32	mega103
Pins	40/44	40/44	40/44	64
Flash	16 KB	16 KB	32 KB	128 KB
SRAM	1 KB	1 KB	2 KB	4 KB
EEPROM	512 B	512 B	1 KB	2 KB
U(S)ART	2	1	1	1
TWI	1	1	1	-
PWM	4	4	4	4
ADC	-	8@10-bit	8@10-bit	8@10-bit
RTC	Yes	Yes	Yes	Yes
JTAG/OCD	-	-	Yes	-
Self Program	Yes	Yes	Yes	-
HW MULT	Yes	Yes	Yes	-
Brown Out	Yes	Yes	Yes	-
Samples	Now	Now	Now	Now
Production	Now	Now	Now	Now

AVR Families: Mega

	mega8	mega16	mega32	mega64	mega128
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	4 KB
U(S)ART	1	1	1	2	2
TWI	1	1	1	1	1
PWM	3	4	4	8	8
ADC	8@10-bit	8@10-bit	8@10-bit	8@10-bit	8@10-bit
RTC	Yes	Yes	Yes	Yes	Yes
JTAG/OCD	-	Yes	Yes	Yes	Yes
Self Program	Yes	Yes	Yes	Yes	Yes
HW MULT	Yes	Yes	Yes	Yes	Yes
Brown Out	Yes	Yes	Yes	Yes	Yes
Samples	Now	Q1/02	Q2/02	Q2/02	Now
Production	Q1/02	Q2/02	Q2/02	Q2/02	Q1/02

AVR Families: Classic

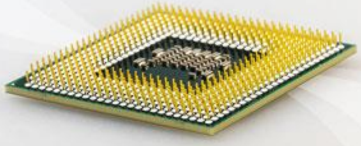


Table 1-2: Some Members of the Classic Family

Part Num.	Code ROM	Data RAM	Data EEPROM	I/O pins	ADC	Timers	Pin numbers & Package
AT90S2313	2K	128	128	15	0	2	SOIC20, PDIP20
AT90S2323	2K	128	128	3	0	1	SOIC8, PDIP8
AT90S4433	4K	128	256	20	6	2	TQFP32, PDIP28

AVR Families: Special Purposes

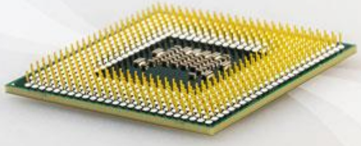
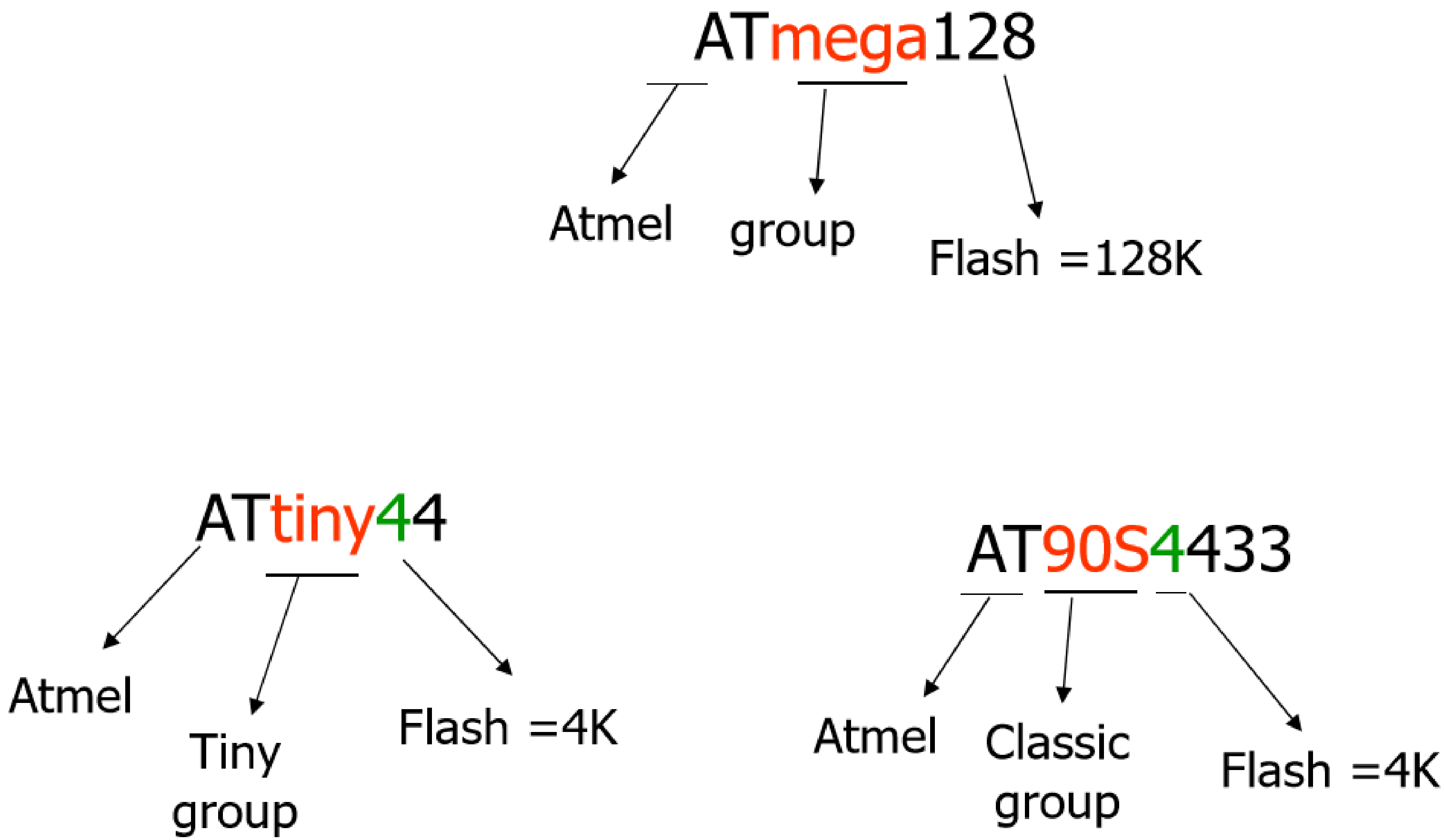
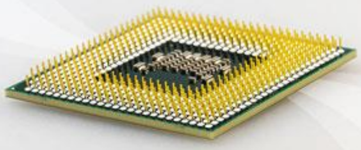


Table 1-6: Some Members of the Special purpose Family							
Part Num	**Code ROM**	**Data RAM**	**Data EEPROM**	**Max I/O pins**	**Special Capabilities**	**Timers**	**Pin numbers & Package**
AT90CAN128	128K	4K	4K	53	CAN	4	LQFP64
AT90USB1287	128K	8K	4K	48	USB Host	4	TQFP64
AT90PWM216	16K	1K	0.5K	19	Advanced PWM	2	SOIC24
ATmega169	16K	1K	0.5K	54	LCD	3	TQFP64,MLF64

Atmel AVR Part Numbers



Atmel AVR ATMEGA32

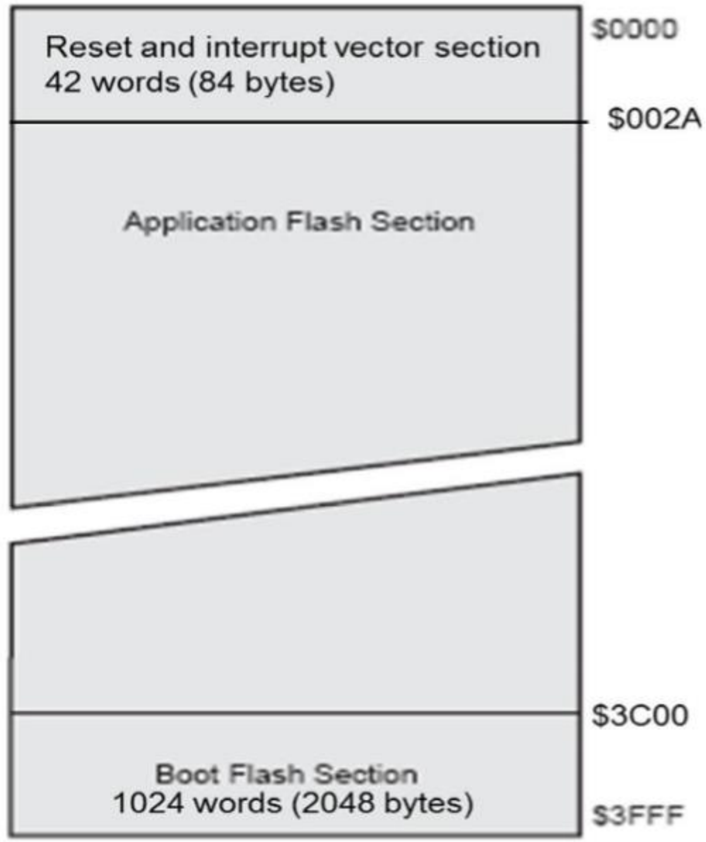


- 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- 32KBytes In-System Programmable Flash
- 1024Bytes EEPROM
- 2Kbytes Internal SRAM
- 32 × 8-bit General Purpose Working Registers
- Two Addressing Modes
- On-chip 2-cycle Multiplier
- Two 8-bit Timer/Counters and one 16-bit Timer/Counters
- Two External Interrupts
- Four PWM Channels
- 8-channel (Multiplexed input) 10-bit A/D converter

ATMega32 Memory

- **32KB Flash Program Memory:** Used to store program code – Memory contents retained when power is off (non-volatile) – Fast to read but slow to write – Can only write entire “blocks” of memory at a time – organized in 16-bit words (16K Words)
- **1KB EEPROM:** For persistent data storage – Memory contents are retained when power is off (non-volatile) – Fast read and slow write – Can write individual bytes
- **2KB SRAM:** For temporary data storage – Memory is lost when power is shut off (volatile) – Fast read and write

ATMEGA32 Flash Memory



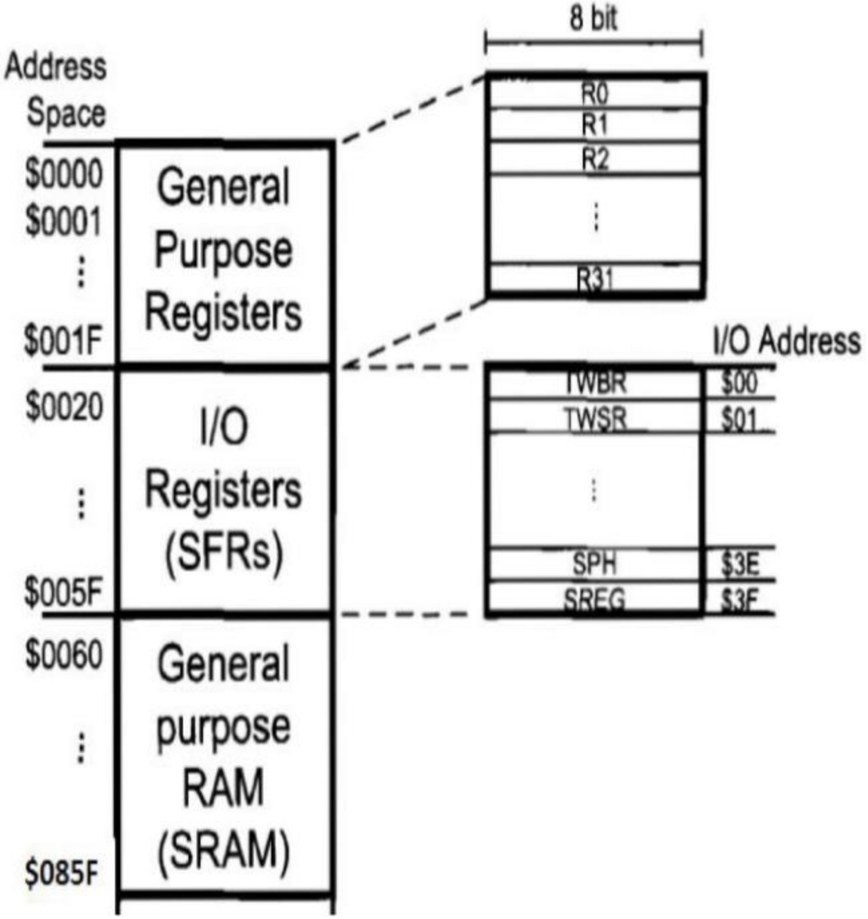
ATMEGA32 Data Memory

The data memory is composed of three parts:

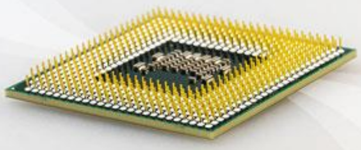
GPRs: General Purpose Registers

SFRs: Special Function Registers

Internal data SRAM

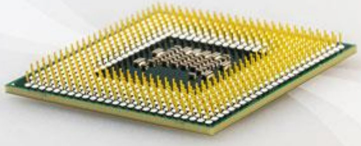


ATMEGA32 Registers



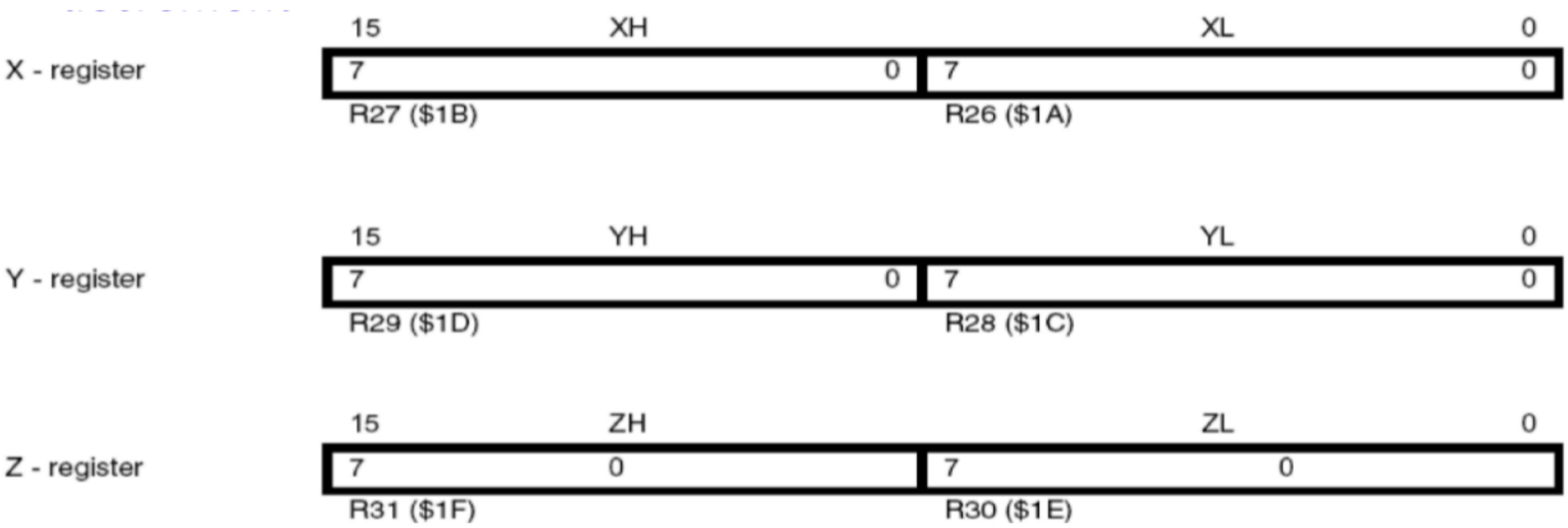
Name	Label	Number	Size	Function
General Purpose Working Register	R0 – R31	32	8 bit	Store data
Address Pointer	X, Y, Z	3	16 bit	Store address pointer
Stack Pointer	SP	1	16 bit	Store a pointer to a group of data know as the stack
Program Counter	PC	1	14 bit	Contains the address of the next instruction to fetch and execute
Status Register	SREG	1	8 bit	Contains information on the results of the last instruction

ATMEGA32 Registers: GPRs



7 0	Addr.	
R0	$00	
R1	$01	
R2	$02	
...		
R13	$0D	
R14	$0E	
R15	$0F	
R16	$10	
R17	$11	
...		
R26	$1A	X-register Low Byte
R27	$1B	X-register High Byte
R28	$1C	Y-register Low Byte
R29	$1D	Y-register High Byte
R30	$1E	Z-register Low Byte
R31	$1F	Z-register High Byte

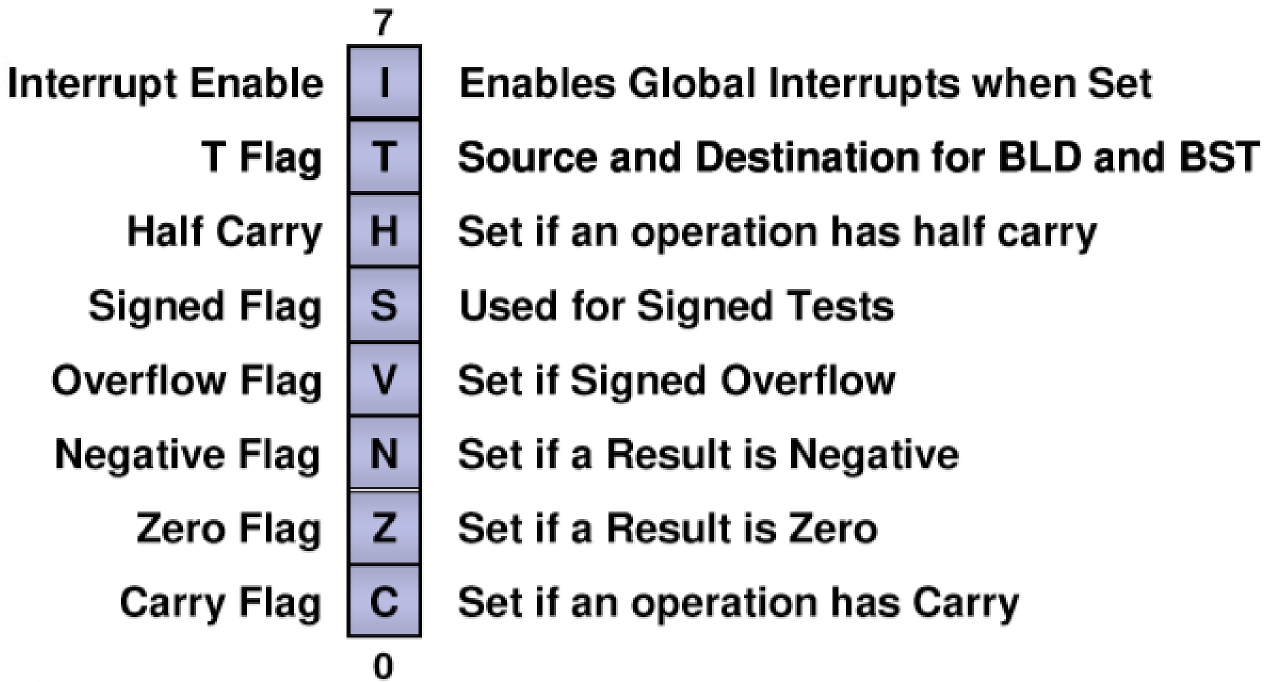
ATMEGA32 Registers: Address Pointer



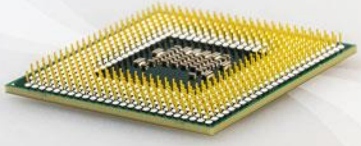
ATMEGA32 Registers: Stack Pointer

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	

ATMEGA32 Registers: Status Register



ATMEGA32 Special Function Registers



Address I/O	Address Mem.	Name
$00	$20	TWBR
$01	$21	TWSR
$02	$22	TWAR
$03	$23	TWDR
$04	$24	ADCL
$05	$25	ADCH
$06	$26	ADCSRA
$07	$27	ADMUX
$08	$28	ACSR
$09	$29	UBRRL
$0A	$2A	UCSRB
$0B	$2B	UCSRA
$0C	$2C	UDR
$0D	$2D	SPCR
$0E	$2E	SPSR
$0F	$2F	SPDR
$10	$30	PIND
$11	$31	DDRD
$12	$32	PORTD
$13	$33	PINC
$14	$34	DDRC
$15	$35	PORTC

Address I/O	Address Mem.	Name
$16	$36	PINB
$17	$37	DDRB
$18	$38	PORTB
$19	$39	PINA
$1A	$3A	DDRA
$1B	$3B	PORTA
$1C	$3C	EECR
$1D	$3D	EEDR
$1E	$3E	EEARL
$1F	$3F	EEARH
$20	$40	UBRRC
		UBRRH
$21	$41	WDTCR
$22	$42	ASSR
$23	$43	OCR2
$24	$44	TCNT2
$25	$45	TCCR2
$26	$46	ICR1L
$27	$47	ICR1H
$28	$48	OCR1BL
$29	$49	OCR1BH
$2A	$4A	OCR1AL

Address I/O	Address Mem.	Name
$2B	$4B	OCR1AH
$2C	$4C	TCNT1L
$2D	$4D	TCNT1H
$2E	$4E	TCCR1B
$2F	$4F	TCCR1A
$30	$50	SFIOR
$31	$51	OCDR
		OSCCAL
$32	$52	TCNT0
$33	$53	TCCR0
$34	$54	MCUCSR
$35	$55	MCUCR
$36	$56	TWCR
$37	$57	SPMCR
$38	$58	TIFR
$39	$59	TIMSK
$3A	$5A	GIFR
$3B	$5B	GICR
$3C	$5C	OCR0
$3D	$5D	SPL
$3E	$5E	SPH
$3E	$5E	SREG

C-Like Addressing Modes

Auto Increment/Decrement:

C Source:

```
unsigned char *var1, *var2;
*var1++ = *--var2;
```

Generated code:

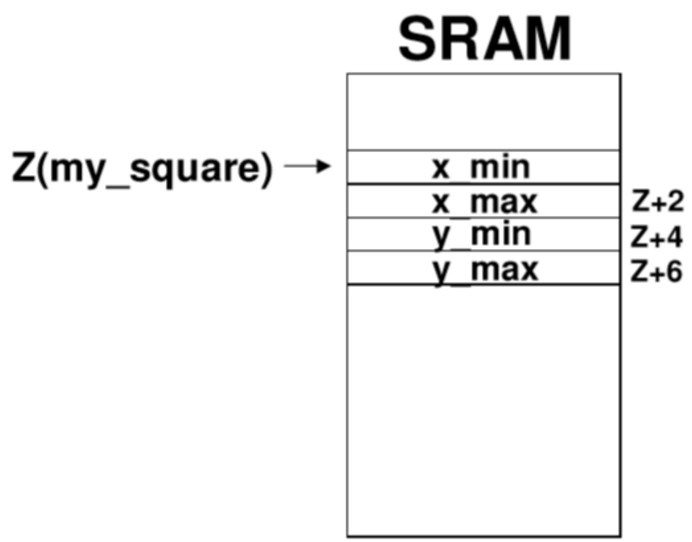
```
LD      R16,-X
ST      Z+,R16
```

C-Like Addressing Modes

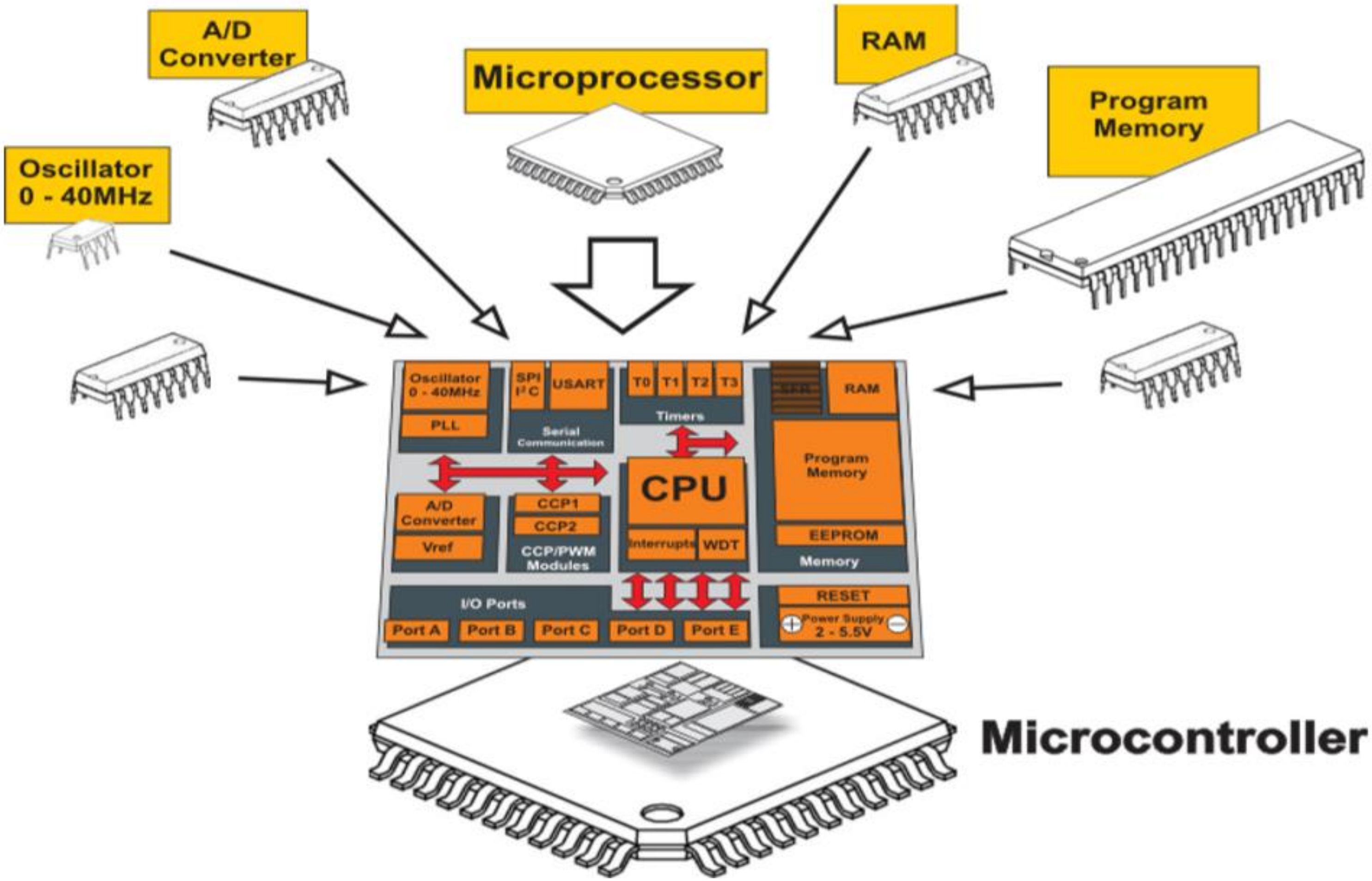
Indirect with Displacement:

- Efficient for accessing arrays and structs
- Autos placed on Software Stack

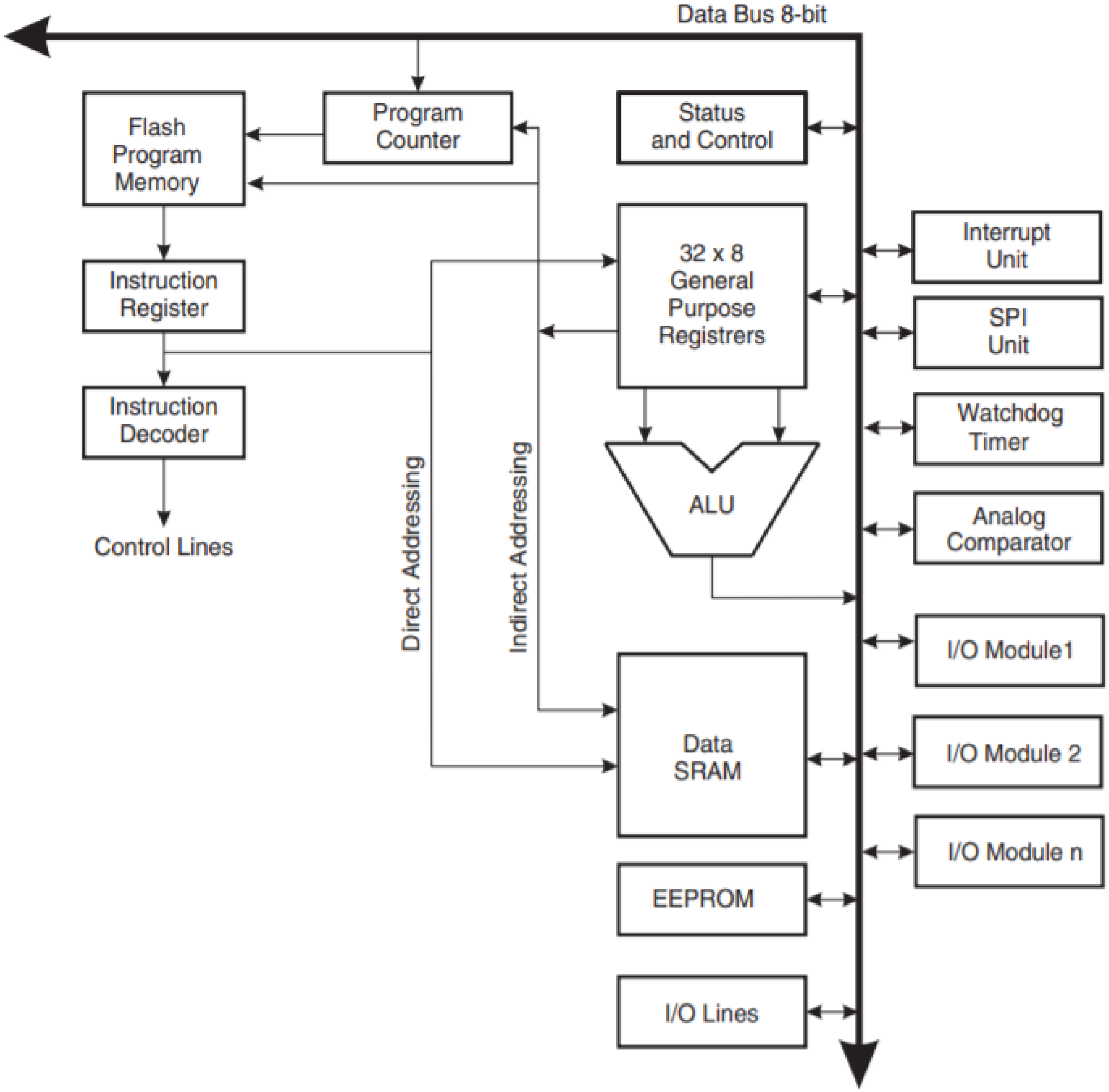
```
Struct square
{
    int x_min;
    int x_max;
    int y_min;
    int y_max;
}my_square;
```



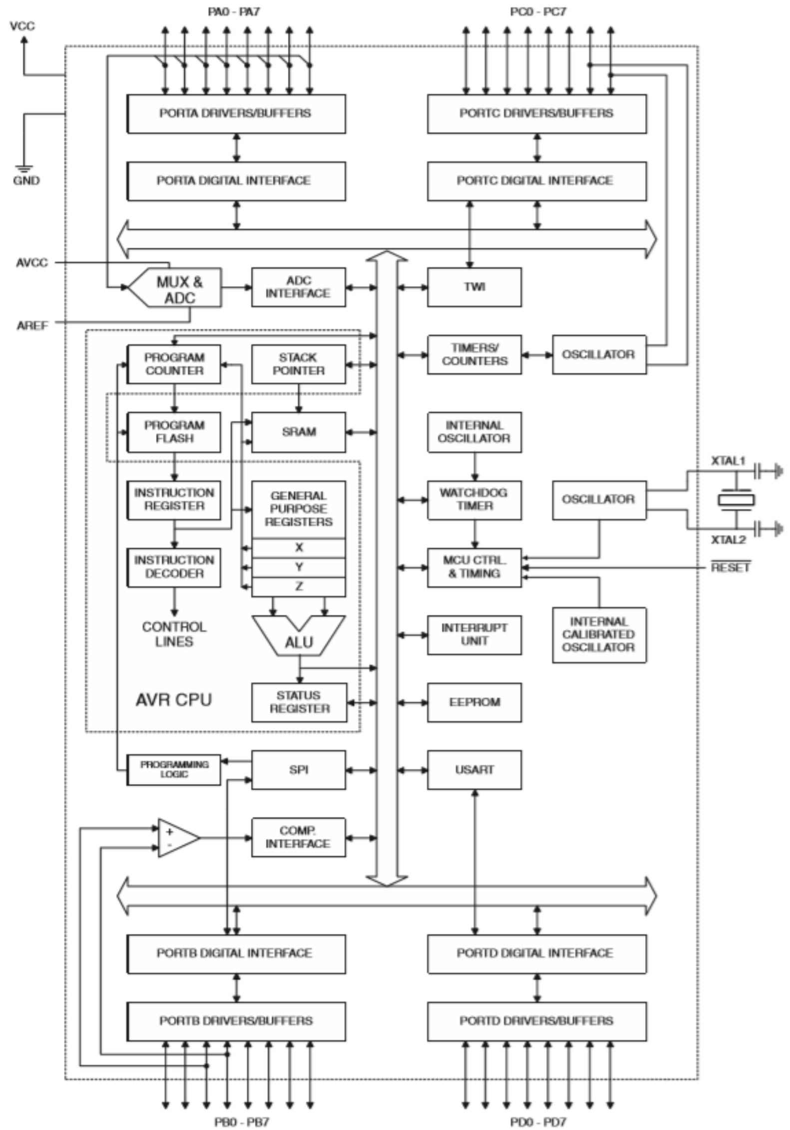
Atmel AVR Structure



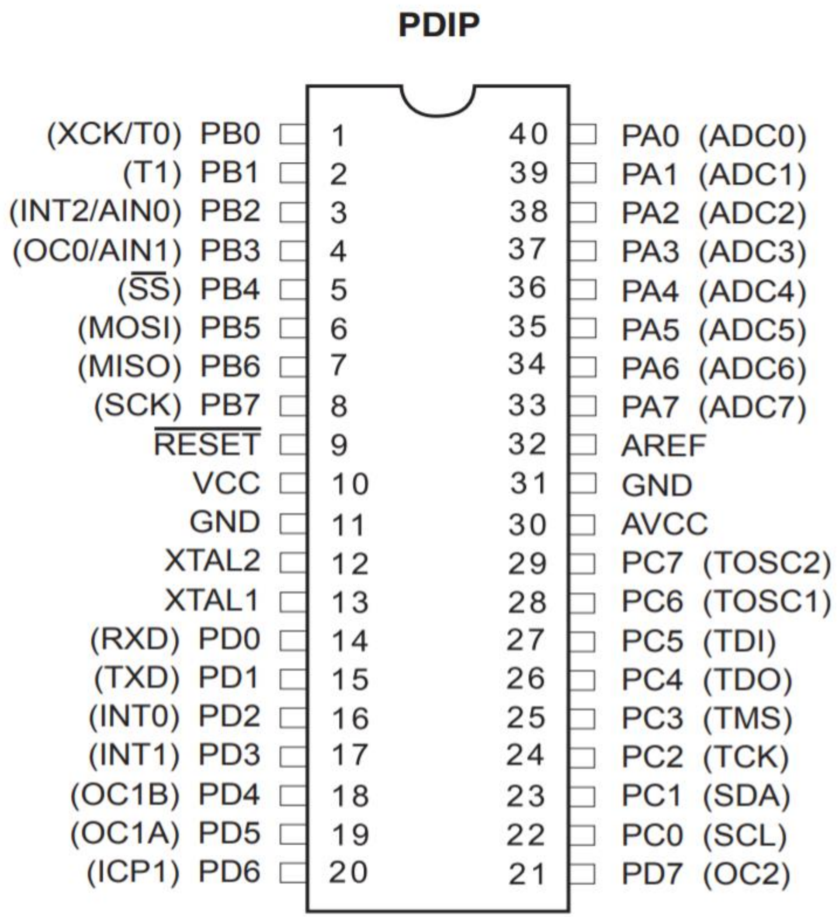
Atmel AVR Architecture



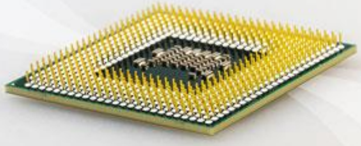
ATMega32 Block Diagram



ATMEGA32 Pin Map

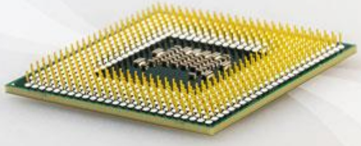


ATMEGA32 Pin Map



VCC	Digital supply voltage.
GND	Ground.
Port A (PA7..PA0)	Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.
Port B (PB7..PB0)	Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port B also serves the functions of various special features of the ATmega32 as listed on page 57.

ATMEGA32 Pin Map



Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

The TD0 pin is tri-stated unless TAP states that shift out data are entered.

Port C also serves the functions of the JTAG interface and other special features of the ATmega32 as listed on page 60.

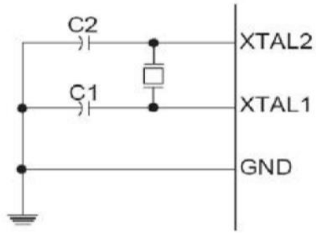
Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega32 as listed on page 62.

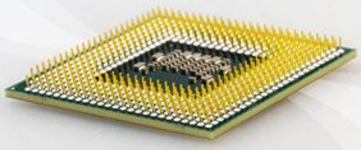
ATMEGA32 Pin Map

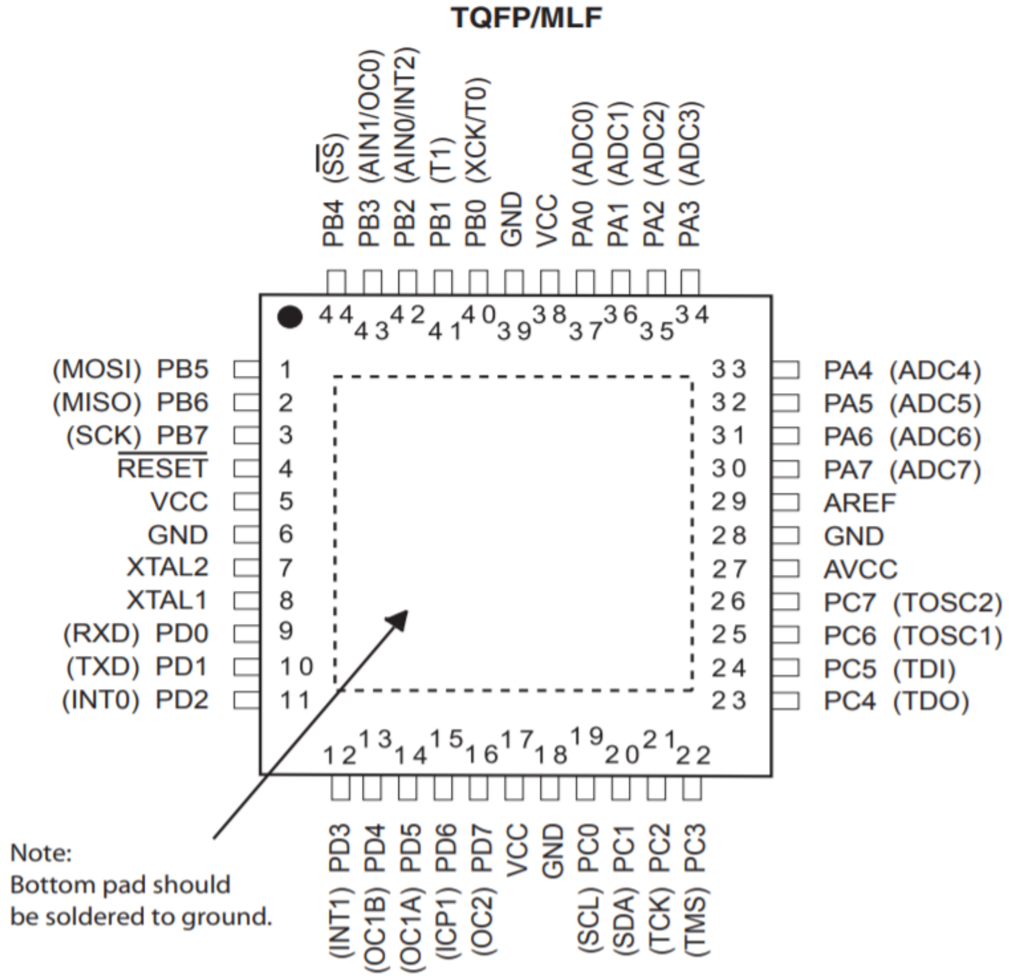
$\overline{\text{RESET}}$	Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 37. Shorter pulses are not guaranteed to generate a reset.
XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2	Output from the inverting Oscillator amplifier.



AVCC	AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.
AREF	AREF is the analog reference pin for the A/D Converter.

ATMEGA32 Pin Map





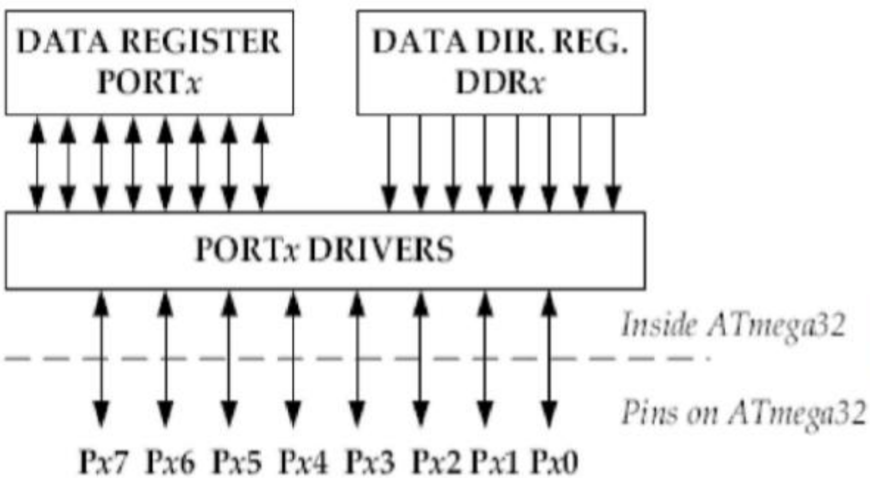
ATMEGA32 Ports Description

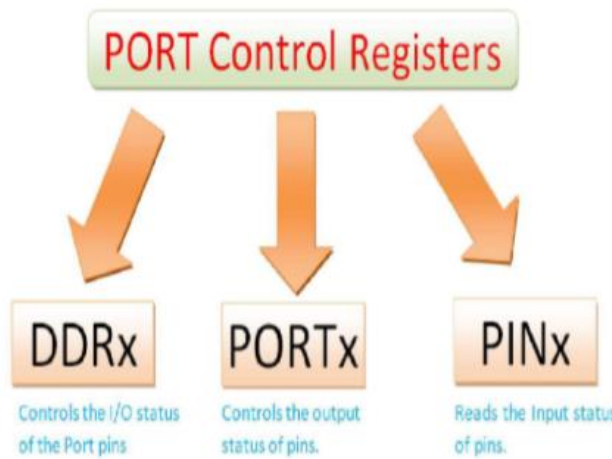
Each Port has three 8-bit Registers associated with it.

DDRx: Data Direction Register for Port x (Read/Write)

PORTx: Data Register for Port x (Read/Write)

PINx: Port Input Pins Register for Port x (Read only)



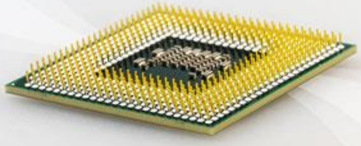


ATMEGA32 Ports Setup

Table 20. Port Pin Configurations

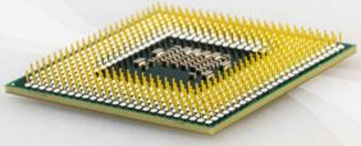
DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if external pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

Starting with a Microcontroller



1. Programmers, Development Boards, Educational Kits, …
2. Compiler
3. Programming Language

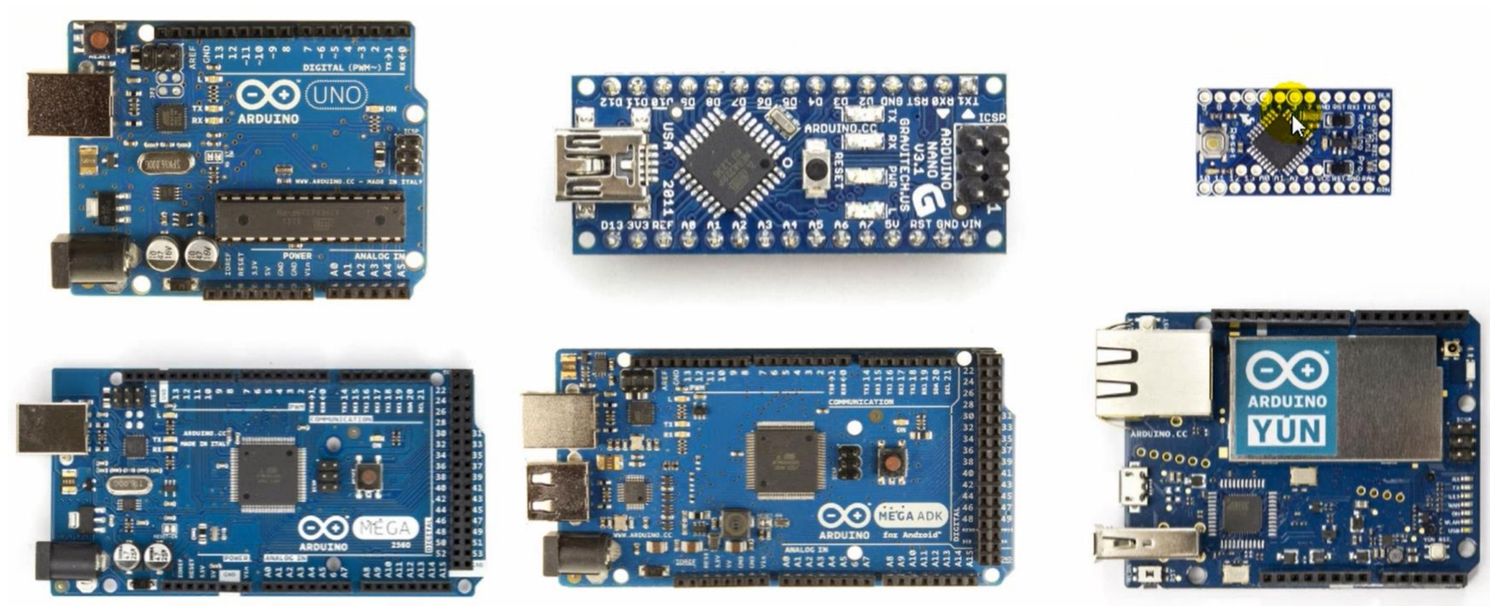
Starting with a Microcontroller



1. Programmers, Development Boards, Educational Kits, ...

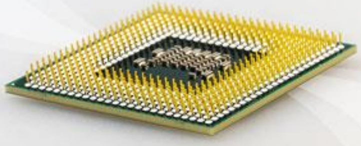
Programmer

Arduino Development Boards



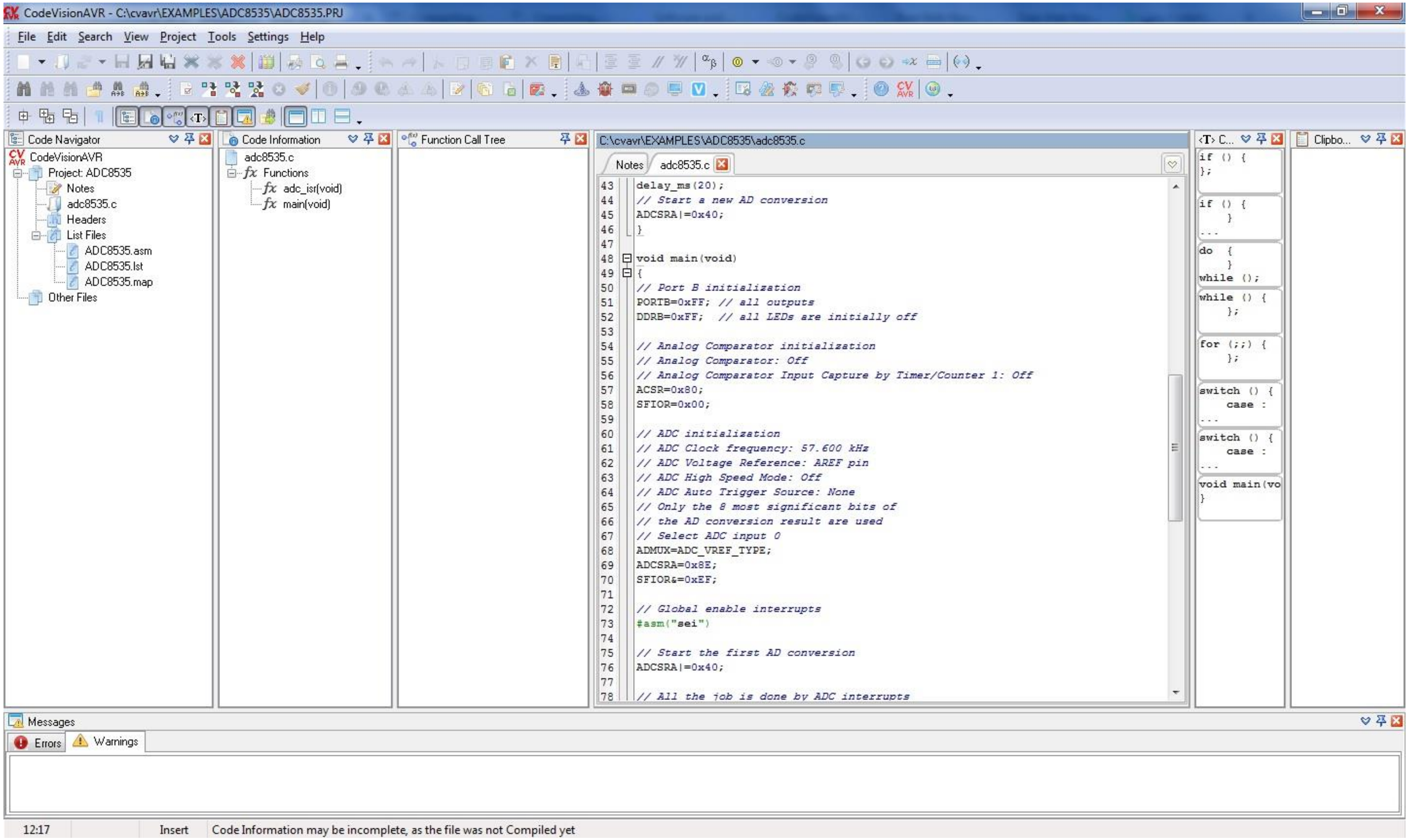
2. Compiler

Compilers

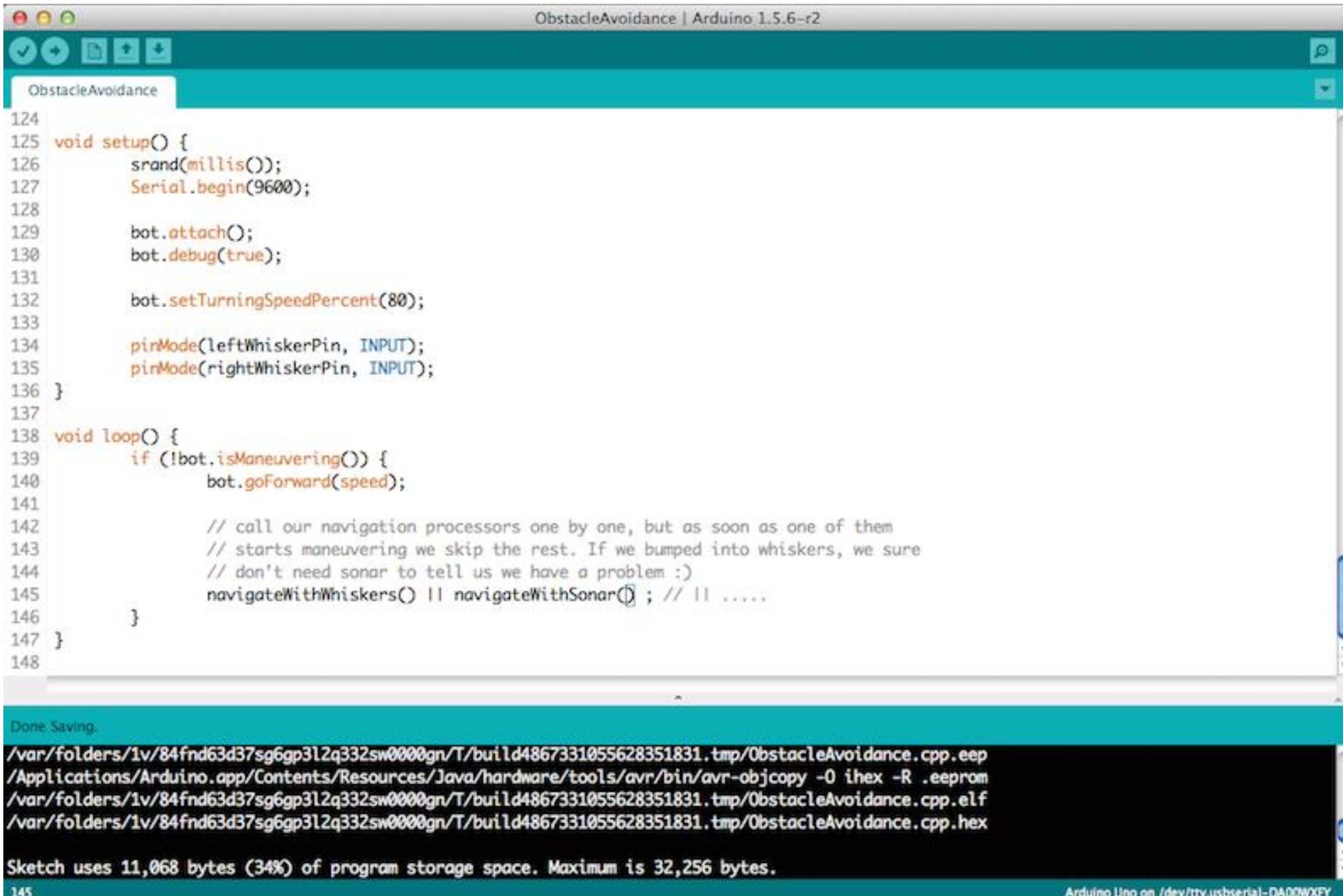


- AVRStudio
- CodeVisionAVR
- Arduino IDE
- Mikro Pro
- Image Craft
- GCC Port
- WinAVR
- IAR
- …

CodeVisionAVR



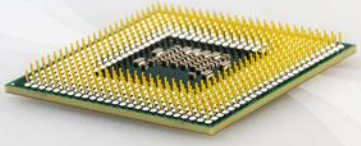
Arduino IDE


ObstacleAvoidance | Arduino 1.5.6-r2
ObstacleAvoidance
void setup() {
srand(millis());
Serial.begin(9600);
bot.attach();
bot.debug(true);
bot.setTurningSpeedPercent(80);
pinMode(leftWhiskerPin, INPUT);
pinMode(rightWhiskerPin, INPUT);
}
void loop() {
if (!bot.isManeuvering()) {
bot.goForward(speed);
// call our navigation processors one by one, but as soon as one of them
// starts maneuvering we skip the rest. If we bumped into whiskers, we sure
// don't need sonar to tell us we have a problem :)
navigateWithWhiskers() || navigateWithSonar() ; // || .....
}
}
Done Saving.
/var/folders/1v/84fnd63d37sg6gp3l2q332sw0000gn/T/build4867331055628351831.tmp/ObstacleAvoidance.cpp.eep
/Applications/Arduino.app/Contents/Resources/Java/hardware/tools/avr/bin/avr-objcopy -O ihex -R .eeprom
/var/folders/1v/84fnd63d37sg6gp3l2q332sw0000gn/T/build4867331055628351831.tmp/ObstacleAvoidance.cpp.elf
/var/folders/1v/84fnd63d37sg6gp3l2q332sw0000gn/T/build4867331055628351831.tmp/ObstacleAvoidance.cpp.hex
Sketch uses 11,068 bytes (34%) of program storage space. Maximum is 32,256 bytes.
145
Arduino Uno on /dev/tty.usbserial-DA00WXFY

Starting with a Microcontroller

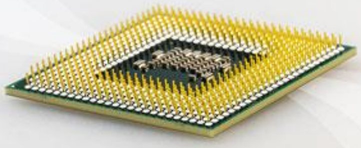
Programming Language

Supported Languages



- Assembly
- C
- C++
- Basic
- Pascal

A Small C Function



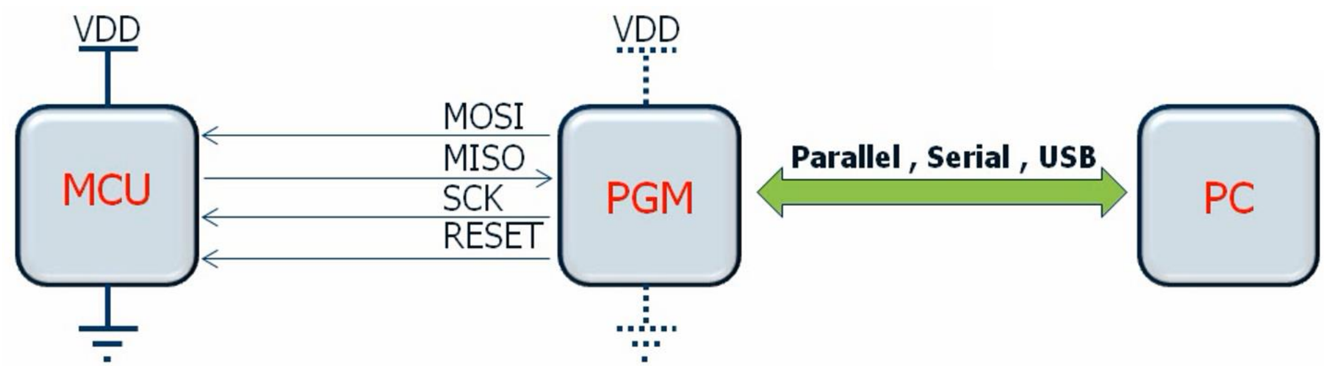
```c
/* Return the maximum value of a table of 16 integers */

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

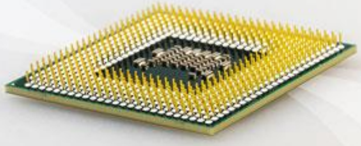
```
; 7.    for (a=0;a<16;a++)
        LDI     R18,LOW(0)
        LDI     R19,128
        CLR     R22
?0001:
        CPI     R22,LOW(16)
        BRCC    ?0000
; 8.      {
; 9.        if (array[a]>maximum)
        MOV     R30,R22
        CLR     R31
        LSL     R30
        ROL     R31
        ADD     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
```

Code size: 46 bytes,   **Execution time:** 335 Cycles

ISP Programmer

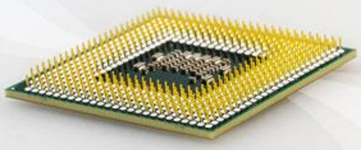


Exercises



1. LED Blinker
2. Full Adder
3. Seven Segment BCD Counter Using 4511 IC
4. Character LCD
5. Pulse Width Modulator (PWM)

Full Range of Development Tools



Evaluation Tools:

- STK500 and AVR Studio

Total Cost **$79**

Low Cost Tools:

- STK500 and AVR Studio
- ICE / JTAGICE
- Imagecraft / CodeVisionAVR / GNU

Total Cost **< $500**

High Performance Tools:

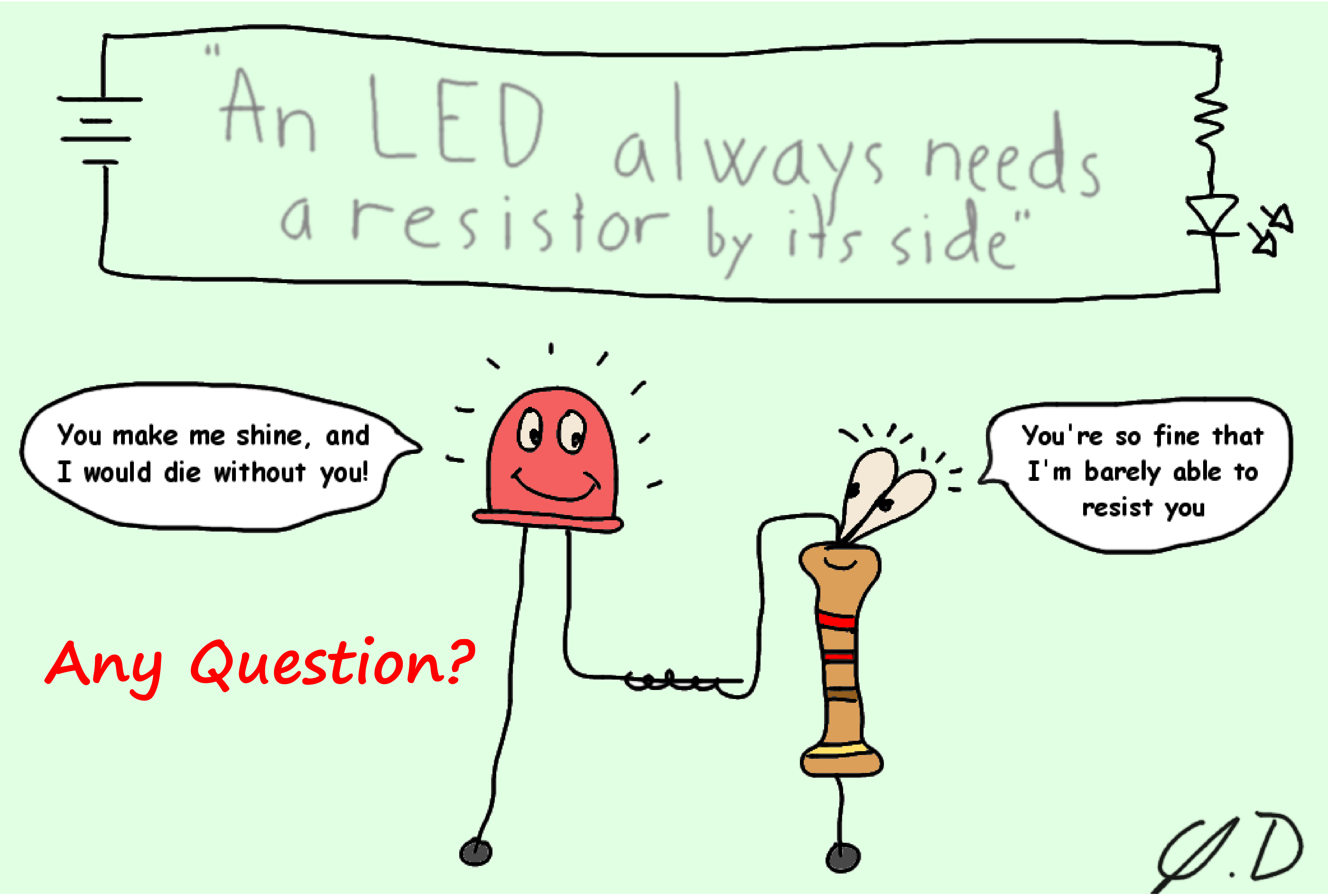
- STK500 and AVR Studio
- ICE30 / ICE10 / ICEPRO
- IAR C / IAR C++

Total Cost **~ $7100**

Cheapest Tools Available for Students:

- Arduino IDE
- Arduino Boards

Total Cost **Starting From $8**


"An LED always needs a resistor by it's side"
You make me shine, and I would die without you!
You're so fine that I'm barely able to resist you
Any Question?


END...