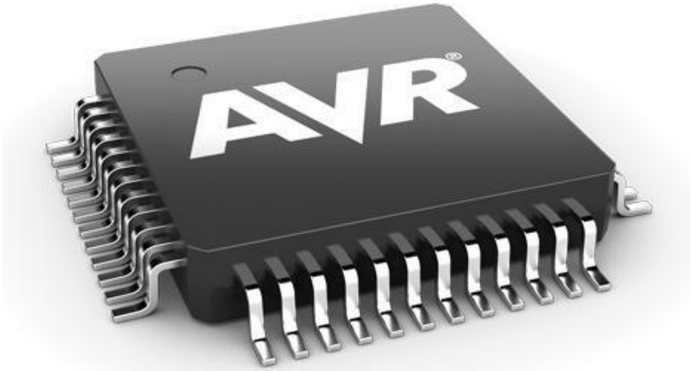


MICROCHIP

Welcome to
1DT301, Computer Technology
autumn 2021!

1DT301, Computer Technology

- Programming in Assembly and C
- STK500 or STK600
- Lectures ~12 + 4
- Laboratory work (6 labs).
- Laboratory work is mandatory
- Written exam

- Studying remotely

Linnæus University
Sweden

Planning, 1DT301, Computer Technology, autumn 2021

Lec	Date:	Time	Room	Content	Lab
F1	2021-09-06	10^{15}-12^{00}	Online	Introduction, Course start Repetition of binary and hexadecimal numbers, two's complement form.	None.
F2	2021-09-08	13^{15}-15^{00}	Online	Experimental environment, STK600, How to use ports. Assembly programming.	
F3	2021-09-10	10^{15}-12^{00}	Online	Subroutine call, function of the STACK. **Introduction to lab1.**	
F4	2021-09-13	10^{15}-12^{00}	Online	Using register pairs, subroutine call, random generator.	Lab 1. How to use the PORTs. Digital input/output.
F5	2021-09-14	13^{15}-15^{00}	Online	Atmel AVR Assembler. Jumping and branching.	
F6	2021-09-17	10^{15}-12^{00}	Online	Using interrupts. **Introduction to lab2.**	
F7	2021-09-20	10^{15}-12^{00}	Online	Using timers and serial communication, USART.	Lab 2. Subroutines
C1	2021-09-24	10^{15}-12^{00}	Online	Introduction to C-programming language. **Introduction to lab3.**	
F8	2021-09-27	10^{15}-12^{00}	Online	Interfacing to Hitachi display.	Lab 3. Interrupt routines
C2	2021-10-01	10^{15}-12^{00}	Online	Introduction to C-programming language. **Introduction to lab4.**	
F9	2021-10-04	10^{15}-12^{00}	Online	Disassembler, instructions op-code.	Lab 4. Timer and UART.
C3	2021-10-08	10^{15}-12^{00}	Online	Introduction to C-programming language. **Introduction to lab5.**	
F10	2021-10-11	10^{15}-12^{00}	Online	Addressing modes, program examples.	Lab 5. LCD-Display.
C4	2021-10-15	10^{15}-12^{00}	Online	Introduction to C-programming language. **Introduction to lab6.**	
F11	2021-10-18	10^{15}-12^{00}	Online	Calculations with multiply, program examples.	Lab 6. CyberTech Wall Display and C-programming.
F12	2021-10-22	10^{15}-12^{00}	Online	Repetition and program examples.	

Manuals and free on-line books:

1. doc2466_ATmega16.pdf - 8-bit Atmel Microcontroller with 4K/128K/256K Bytes In-System Programmable Flash
2. Instruction set.pdf – ATMEL 8-bit AVR Instruction Set
3. doc1022_AVR_Assembler_user_guide.pdf - AVR Assembler User Guide
4. JHD202C_display_2x20.pdf
5. Beginners Introduction to the Assembly Language of ATMEL-AVR-Microprocessors (English, Free on-line book), www.avr-asm-download.de/beginner_en.pdf

Text books:

1. Programming and Interfacing ATMEL's AVRs, 1st Edtion, chapter 2, page 29 – 166 (English)
2. Mikroprocessorteknik, Per Foyer, Studentlitteratur. ISBN 91-44-03876-3 (Swedish)

Schedule

https://cloud.timeedit.net/lnu/web/schema2/riqvoQ0qY6XZ9QQyYp197ZQ061Zcw56QQZZ6705Q9Yo.html

Communication channels

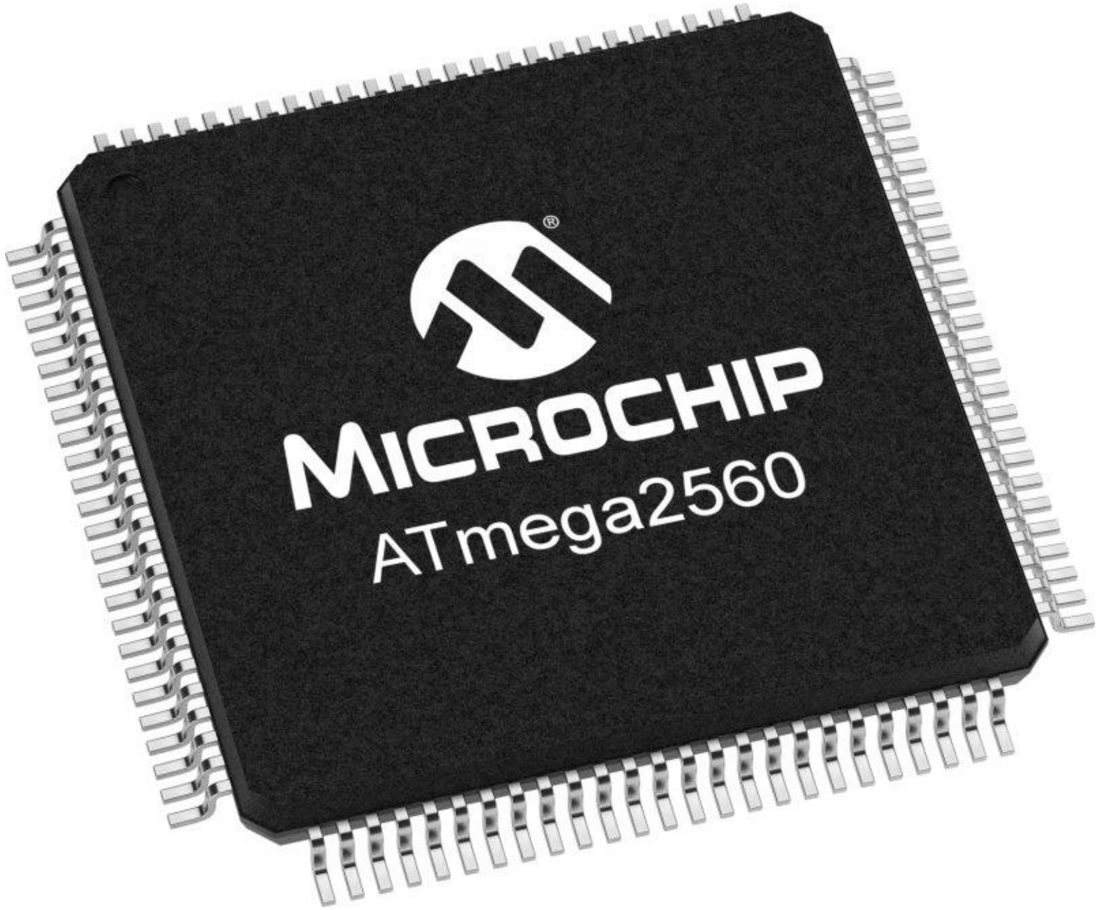
MyMoodle is the official channel for communicating messages.

Slack (#1DT301-comp-tech) is the unofficial channel.

Are you studying remotely?

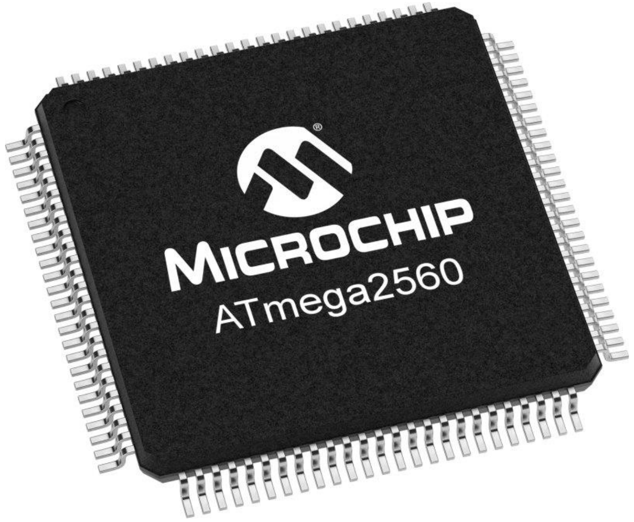
I need to figure out which of You are studying remotely. So, if You are studying remotely send me an email (kala224@lnu.se) where You tell from where.

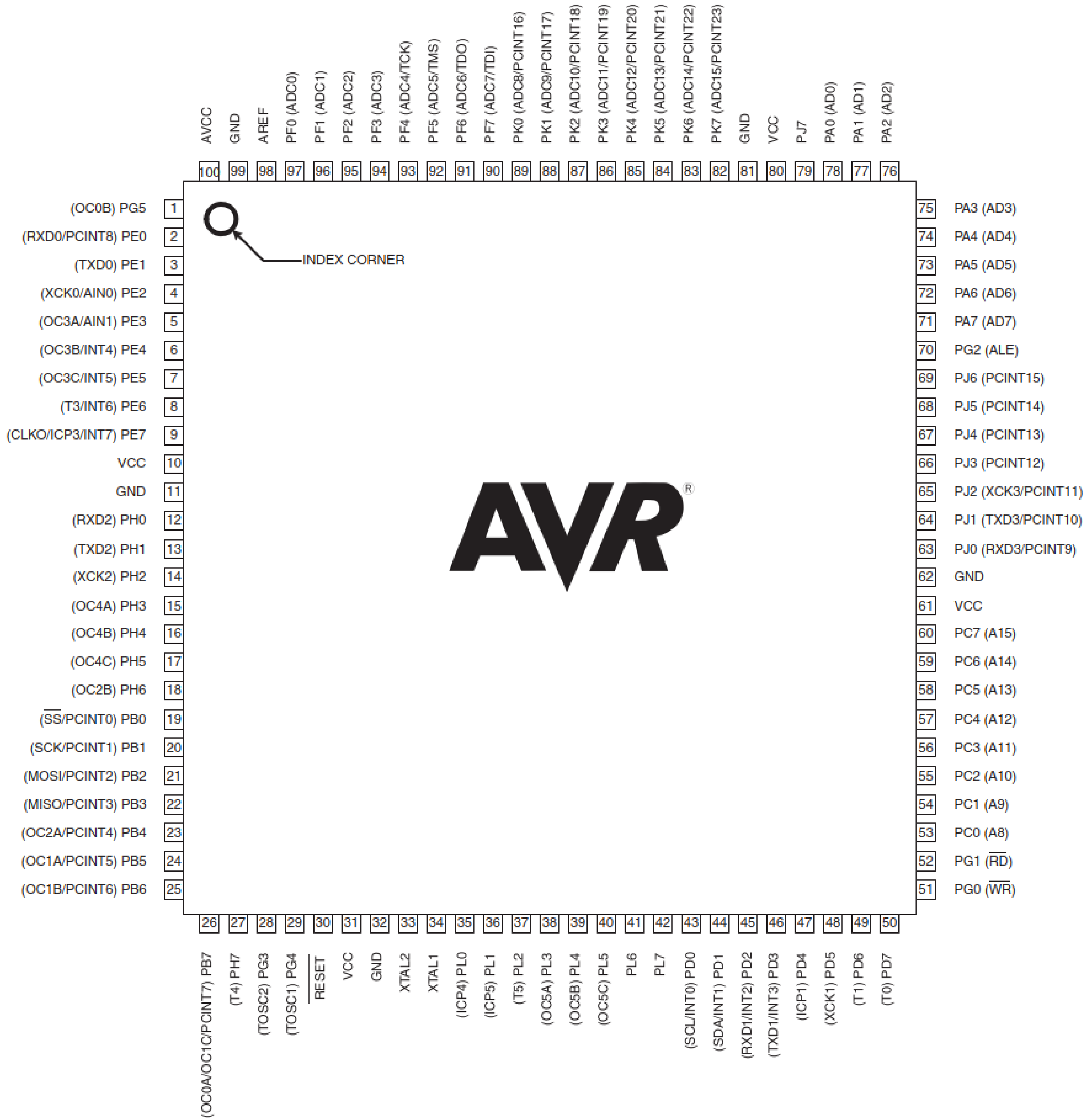
ATMega2560



Max ADC Resolution (bits)	10
Program Memory Size (KB)	256
Capture/Compare/PWM (CCP)	4
Number of Comparators	1
CPU Speed (MIPS/DMIPS)	16
Data EEPROM (bytes)	4096
DigitalTimerQty_16bit	4
Max 8 Bit Digital Timers	2
Ethernet	None
I2C	1
Program Memory Type	Flash
mtrlcntrlinputcapture	4
ADC Channels	16
Low Power	No
Operating Voltage	1.8 - 5.5
outputcomparatorPWM	16
Pin Count	100
SPI	5
Temp Range (°C)	-40 - 85
USART	4

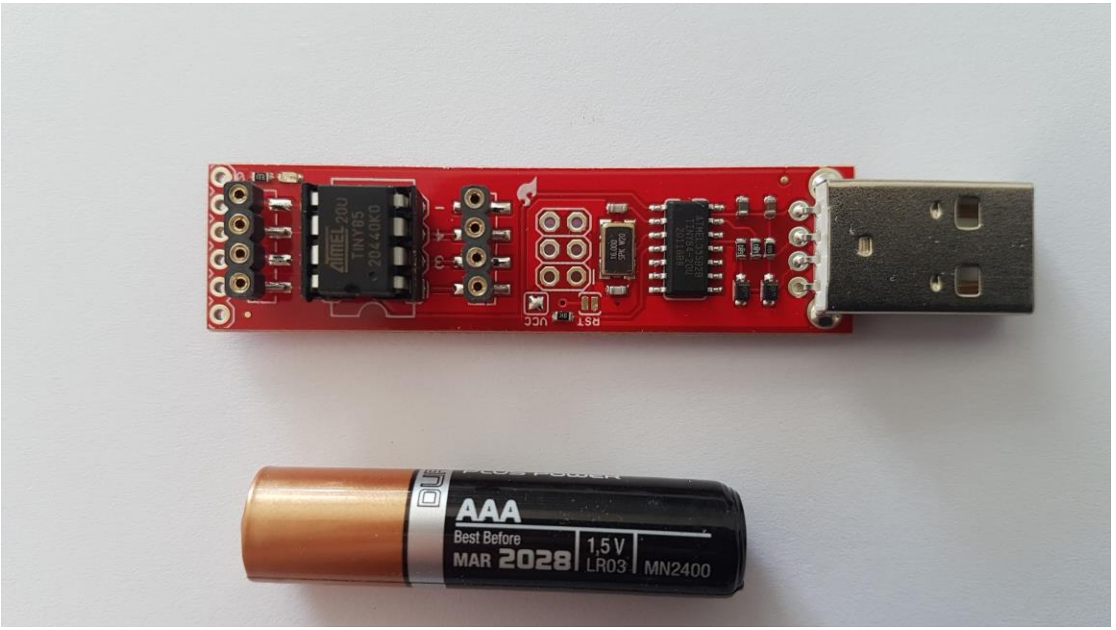
Pin Configuration





STK 600

ATTiny85



Parametrics

Click on a property to perform a parametric search for other products with that property.

Max ADC Resolution (bits)	10
Program Memory Size (KB)	8
Number of Comparators	1
CPU Speed (MIPS/DMIPS)	20
Data EEPROM (bytes)	512
Max 8 Bit Digital Timers	2
Ethernet	None
I2C	1
Program Memory Type	Flash
ADC Channels	4
Low Power	No
Operating Voltage	1.8 - 5.5
outputcomparatorPWM	5
Pin Count	8
SPI	1
Temp Range (°C)	-40 - 85

Architecture

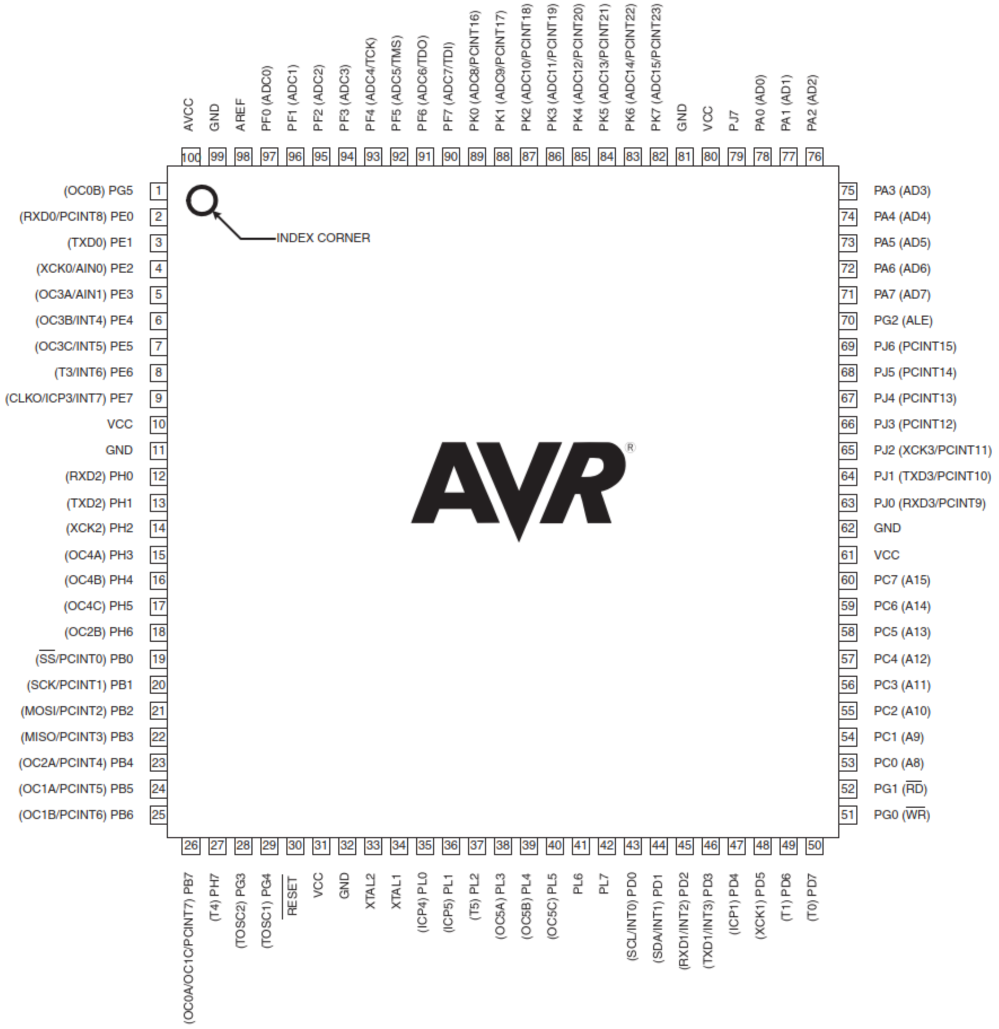
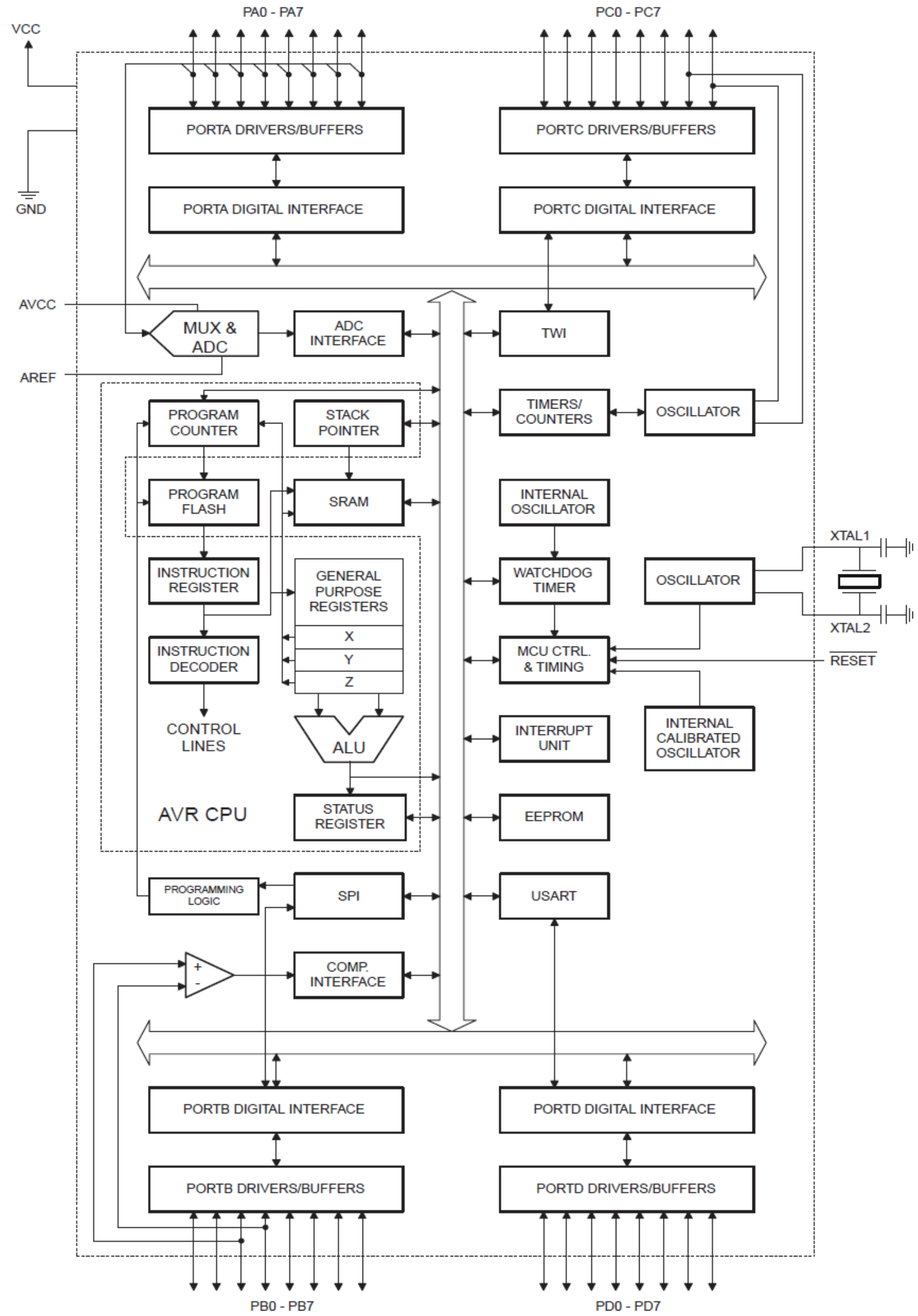
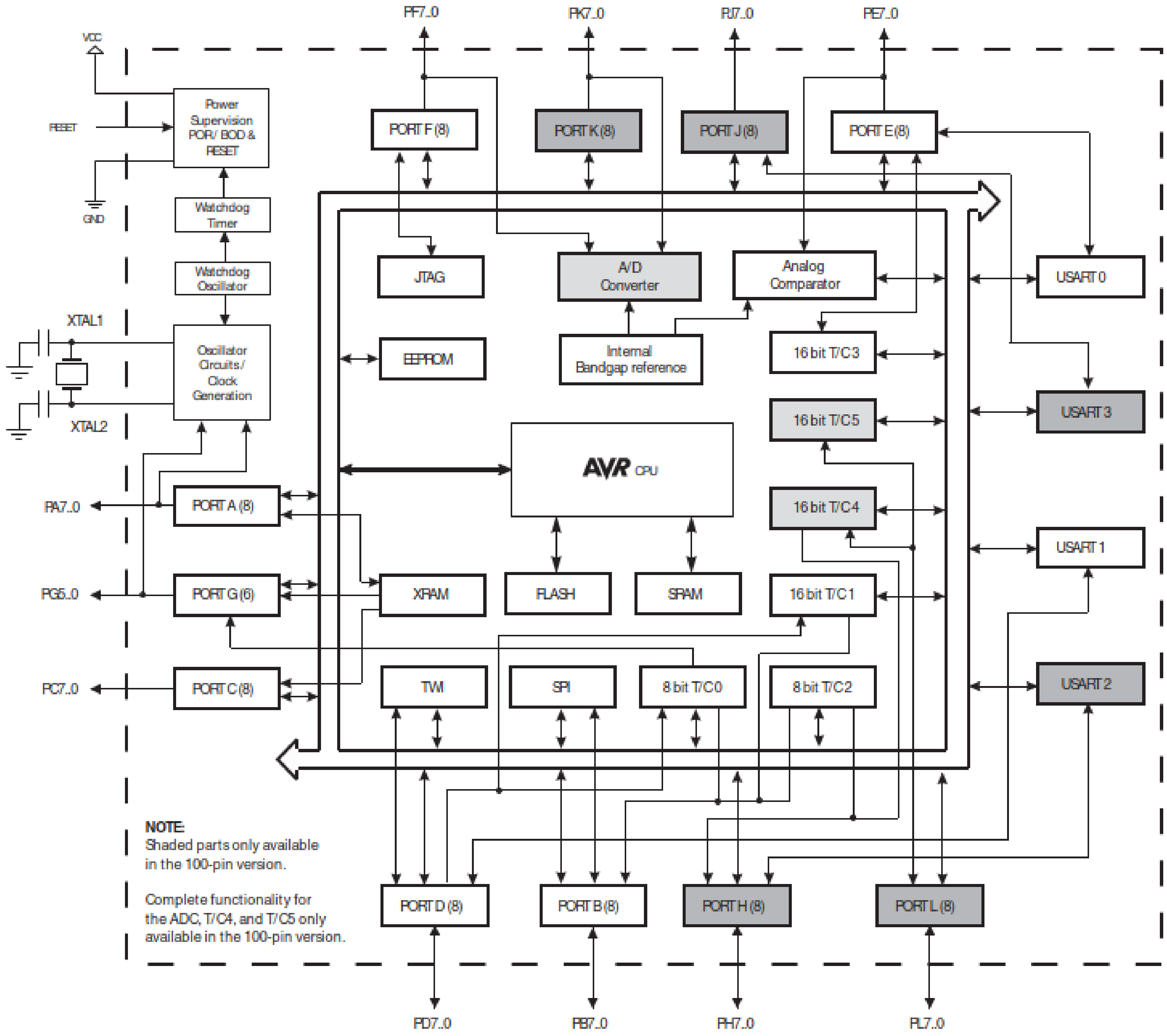


Figure 2. Block Diagram




PF7..0
PK7..0
PJ7..0
PE7..0
VCC
RESET
GND
Power Supervision POR/ BOD & RESET
Watchdog Timer
Watchdog Oscillator
XTAL1
XTAL2
Oscillator Circuits / Clock Generation
PORT F (8)
PORT K (8)
PORT J (8)
PORT E (8)
JTAG
A/D Converter
Analog Comparator
USART 0
EEPROM
Internal Bandgap reference
16 bit T/C3
16 bit T/C5
USART 3
AVR CPU
16 bit T/C4
PA7..0
PORT A (8)
USART 1
PG5..0
PORT G (6)
XRAM
FLASH
SRAM
16 bit T/C1
PC7..0
PORT C (8)
TWI
SPI
8 bit T/C0
8 bit T/C2
USART 2
NOTE:
Shaded parts only available in the 100-pin version.
Complete functionality for the ADC, T/C4, and T/C5 only available in the 100-pin version.
PORT D (8)
PORT B (8)
PORT H (8)
PORT L (8)
PD7..0
PB7..0
PH7..0
PL7..0

Assemby or Assembler language.

Assembly language

From Wikipedia, the free encyclopedia

An **assembly language** (or **assembler language**),[1] often abbreviated **asm**, is any low-level programming language in which there is a very strong correspondence between the instructions in the language and the architecture's machine code instructions.[2] Assembly language may also be called *symbolic machine code*.[3][4]

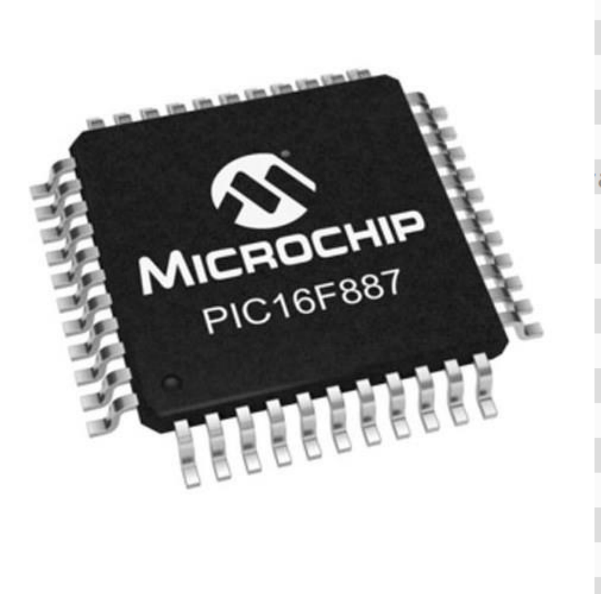
Assembly code is converted into executable machine code by a utility program referred to as an *assembler*. The conversion process is referred to as *assembly*, as in *assembling* the source code. Assembly language usually has one statement per machine instruction (1:1), but comments and statements that are assembler directives,[5] macros,[6][1] and symbolic labels of program and memory locations are often also supported.

Each assembly language is specific to a particular computer architecture and sometimes to an operating system.[7] However, some assembly languages do not provide specific syntax for operating system calls, and most assembly languages can be used universally with any

Different CPUs,
different Assembly languages

Assembly (x86)

```
; TODO INSERT CONFIG CODE HERE USING CONFIG BITS GENERATOR
#INCLUDE <P16F84A.INC>
RES_VECT  CODE    0x0000            ; processor reset vector
    GOTO    START                   ; go to beginning of program

; TODO ADD INTERRUPTS HERE IF USED

CBLOCK 0x0C
 COUNT1
 COUNT2
ENDC

MAIN_PROG CODE                                                   am

 START
    BSF STATUS, RP0
    MOVLW 0xFE
    MOVWF TRISB
    BCF STATUS, RP0

 MAIN
    BSF PORTB,0
    CALL DELAY
    BCF PORTB,0
    CALL DELAY
    GOTO MAIN

```




```
//
//asynchronous 8n1 serial transmi
//
#define TX_PIN 6            //TX p

.global SerialPutChar

SerialPutChar:
        push r16
        push r17
        ldi  r16, 10         ;1 start + 8 data + 1 stop bits (bit count)
        com  r24             ;Invert everything (r24 = byte to xmit)
        sec                  ;Start bit
putchar0:
        brcc putchar1        ;If carry set
        cbi  _SFR_IO_ADDR(PORTA), TX_PIN ;send a '0'
        rjmp putchar2        ;else
putchar1:
        sbi  _SFR_IO_ADDR(PORTA), TX_PIN ;send a '1'
        nop
putchar2:
        rcall UARTDelay      ;1/2 bit delay +
        rcall UARTDelay      ;1/2 bit delay = 1bit delay
        lsr   r24            ;Get next bit
        dec   r16            ;If not all bits sent
        brne  putchar0       ;send next
                             ;else
        pop   r17
```

Microchip has purchased Atmel in 2015.
Microcip is thus selling AVR micro controllers.

Documentation. doc2549_ATmega2560.

Most important document.
Includes almost everything about the controller that you have to know for programming it.
Example:
Pin lay-out of CPU
Different registers
Memory configuration etc.

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256KBytes of In-System Self-Programmable Flash
  - 4Kbytes EEPROM
  - 8Kbytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix® acquisition
  - Up to 64 sense channels
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages

8-bit Atmel Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash

ATmega640/V
ATmega1280/V
ATmega1281/V
ATmega2560/V
ATmega2561/V

Preliminary

Documentation.
8-bit AVR Instruction Set.

Second most important document.
Includes all instructions and
detailed information about them.

Instruction Set Nomenclature

Status Register (SREG)

SREG:	Status Register
C:	Carry Flag
Z:	Zero Flag
N:	Negative Flag
V:	Two's complement overflow indicator
S:	$N \oplus V$, For signed tests
H:	Half Carry Flag
T:	Transfer bit used by BLD and BST instructions
I:	Global Interrupt Enable/Disable Flag

Registers and Operands

Rd:	Destination (and source) register in the Register File
Rr:	Source register in the Register File
R:	Result after instruction is executed
K:	Constant data
k:	Constant address
b:	Bit in the Register File or I/O Register (3-bit)
s:	Bit in the Status Register (3-bit)
X,Y,Z:	Indirect Address Register (X=R27:R26, Y=R29:R28 and Z=R31:R30)
A:	I/O location address
q:	Displacement for direct addressing (6-bit)

ATMEL®

8-bit AVR®
Instruction Set

Documentation.
M2560def, include file.

This file includes all definitions and register names. With this file included in a program, we can use names instead absolut addresses.

```
                              m2560def
;***** THIS IS A MACHINE GENERATED FILE - DO NOT EDIT ********************
;***** Created: 2011-08-25 20:59 ******* Source: ATmega2560.xml **********
;*************************************************************************
;* A P P L I C A T I O N   N O T E   F O R   T H E   A V R   F A M I L Y
;*
;* Number              : AVR000
;* File Name           : "m2560def.inc"
;* Title               : Register/Bit Definitions for the ATmega2560
;* Date                : 2011-08-25
;* Version             : 2.35
;* Support E-mail      : avr@atmel.com
;* Target MCU          : ATmega2560
;*
;* DESCRIPTION
;* When including this file in the assembly program file, all I/O register
;* names and I/O register bit names appearing in the data book can be used.
;* In addition, the six registers forming the three data pointers X, Y and
;* Z have been assigned names XL - ZH. Highest RAM address for Internal
;* SRAM is also defined
;*
;* The Register names are represented by their hexadecimal address.
;*
;* The Register Bit names are represented by their bit number (0-7).
;*
;* Please observe the difference in using the bit names with instructions
;* such as "sbr"/"cbr" (set/clear bit in register) and "sbrs"/"sbrc"
;* (skip if bit in register set/cleared). The following example illustrates
;* this:
```

Documentation.
m2560def, include file.

Example:
We will use a register which is called DDRD. It is located in this CPU (ATmega2560) on address 0x0a. In a different CPU it can be an other address. By including the file m2560def, we can overcome this problem.

m2560def

```
.equ    TCNT0   = 0x26
.equ    TCCR0B  = 0x25
.equ    TCCR0A  = 0x24
.equ    GTCCR   = 0x23
.equ    EEARH   = 0x22
.equ    EEARL   = 0x21
.equ    EEDR    = 0x20
.equ    EECR    = 0x1f
.equ    GPIOR0  = 0x1e
.equ    EIMSK   = 0x1d
.equ    EIFR    = 0x1c
.equ    PCIFR   = 0x1b
.equ    TIFR5   = 0x1a
.equ    TIFR4   = 0x19
.equ    TIFR3   = 0x18
.equ    TIFR2   = 0x17
.equ    TIFR1   = 0x16
.equ    TIFR0   = 0x15
.equ    PORTG   = 0x14
.equ    DDRG    = 0x13
.equ    PING    = 0x12
.equ    PORTF   = 0x11
.equ    DDRF    = 0x10
.equ    PINF    = 0x0f
.equ    PORTE   = 0x0e
.equ    DDRE    = 0x0d
.equ    PINE    = 0x0c
.equ    PORTD   = 0x0b
.equ    DDRD    = 0x0a
```

Complete Instruction Set Summary

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
		Arithmetic and Logic Instructions			
ADD	Rd, Rr	Add without Carry	Rd ← Rd+ Rr	Z,C,N,V,S,H	1
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1
ADIW	Rd, K	Add Immediate to Word	Rd+1:Rd ← Rd+1:Rd + K	Z,C,N,V,S	2 (1)
SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1
SBIW	Rd, K	Subtract Immediate from Word	Rd+1:Rd ← Rd+1:Rd - K	Z,C,N,V,S	2 (1)
AND	Rd, Rr	Logical AND	Rd ← Rd • Rr	Z,N,V,S	1
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd • K	Z,N,V,S	1
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1
ORI	Rd, K	Logical OR with Immediate	Rd ← Rd v **K**	Z,N,V,S	1
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V,S	1
COM	Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V,S	1
NEG	Rd	Two's Complement	Rd ← $00 - Rd	Z,C,N,V,S,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1
INC	Rd	Increment	Rd ← Rd+ 1	Z,N,V,S	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1

Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
Data Transfer Instructions					
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1 (1)
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LDS	Rd, k	Load Direct from data space	Rd ← (k)	None	2 (1)(4)
LD	Rd, X	Load Indirect	Rd ← (X)	None	2 (2)(4)
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2 (2)(4)
LD	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, Rd ← (X)	None	2 (2)(4)
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2 (2)(4)
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2 (2)(4)
LD	Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1, Rd ← (Y)	None	2 (2)(4)
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2 (1)(4)
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2 (2)(4)
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2 (2)(4)
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2 (2)(4)
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2 (1)(4)
STS	k, Rr	Store Direct to data space	(k) ← Rd	None	2 (1)(4)
ST	X, Rr	Store Indirect	(X) ← Rr	None	2 (2)(4)
ST	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	2 (2)(4)
ST	-X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2 (2)(4)

MOV – Copy Register

Description:

This instruction makes a copy of one register into another. The source register Rr is left unchanged, while the destination register Rd is loaded with a copy of Rr.

Operation:

(i) $Rd \leftarrow Rr$

	Syntax:	Operands:	Program Counter:
(i)	MOV Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode:

0010	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
        mov     r16,r0    ; Copy r0 to r16
        call    check     ; Call subroutine
        ...
check:  cpi     r16,$11   ; Compare r16 to $11
        ...
        ret               ; Return from subroutine
```

Words: 1 (2 bytes)

Cycles: 1

The decimal system.

The decimal number 2019:

$2019 = 2\cdot1000 + 0\cdot100 + 1\cdot10 + 9\cdot1$

$2019 = 2\cdot10^3 + 0\cdot10^2 + 1\cdot10^1 + 9\cdot10^0 = 2\cdot1000 + 0\cdot100 + 1\cdot10 + 9\cdot1 =$

$= 2000 + 0 + 10 + 9 = 2019$

$10^0 = ?$

$100^0 = ?$

$16^0 = ?$

$2^0 = ?$

$\boxed{10^0 = 100^0 = 16^0 = 2^0 = 1}$

Each position has a value, which is the base raised to…: . . . 10^3, 10^2, 10^1, 10^0

The decimal system..

Decimal number with fraction:

$567{,}318 = 5 \cdot 100 + 6 \cdot 10 + 7 \cdot 1 + 3 \cdot 0{,}1 + 1 \cdot 0{,}01 + 8 \cdot 0{,}001 =$

$567{,}318 = 5 \cdot 10^2 + 6 \cdot 10^1 + 7 \cdot 10^0 + 3 \cdot 10^{-1} + 1 \cdot 10^{-2} + 8 \cdot 10^{-3} =$

$= 500 + 60 + 7 + 0{,}3 + 0{,}01 + 0{,}008 = 567{,}318$

The position left of the decimal point has the value the base^0, in this case 10^0.

Positional numeral system, each position has a value:

$\underbrace{\ldots\ 10^2, 10^1, 10^0}_{\text{Integer}}\ ,\ \underbrace{10^{-1}, 10^{-2}, 10^{-3} \ldots}_{\text{Fraction}}$

Ev. decimal point

$10^0 = 100^0 = 16^0 = 2^0 = 1$

Generellt:

$a^0 = 1$ for all a except 0

Binary numbers.

Example, binary number with 8 bits, integer (= 1 Byte):

1	**1**	**0**	**1**	**0**	**1**	**1**	**1**	Binary number
2^7	**2^6**	**2^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**	Positional value, power of 2
			Osv.	$2 \cdot 2 \cdot 2$	$2 \cdot 2$	2	$2^0 = 1$	
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**	Positional value, in decimal

$$1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 =$$

$$1 \cdot 128 + 1 \cdot 64 + 1 \cdot 16 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 215$$

ie:

$1101\ 0111_2 = 215_{10}$

Radix 2 tells the base is 2, Binary number system

Radix 10 tells the base is 10, Decimal system

Binary numbers.

Example, biggest integer value with 8 bits. (= 1 Byte):

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**
2^7	**2^6**	**2^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**

Highest integer with 8 bits.

$1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 =$

$1 \cdot 128 + 1 \cdot 64 + 1 \cdot 32 + 1 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 64 + 1 \cdot 2 + 1 \cdot 1 = 255$

ie:

$1111\ 1111_2 = 255_{10}$

Binary numbers.

Exampel, smallest integer value with 8 bits. (= 1 Byte):

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
2^7	**2^6**	**2^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**

Smallest integer with 8 bits

$$0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 0$$

ie:

$0000\ 0000_2 = 0_{10}$

Binary numbers - fractions.

Example, a binary number with 8 bits, with fractional part (= 1 Byte),
4 bits integer and 4 bits fraction:

1	**1**	**0**	**1**	**0**	**1**	**1**	**1**
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
8	**4**	**2**	**1**	**1/2**	**1/4**	**1/8**	**1/16**

Binary number

Positional value, power of 2.

Positional value, decimal.

What is the positional value, now?

Decimal point, not visible.

$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} =$

$1 \cdot 8 + 1 \cdot 4 + 1 \cdot 1 + 1 \cdot 0{,}25 + 1 \cdot 0{,}125 + 1 \cdot 0{,}0625 = 13{,}4375$

ie:

$1101\ 0111_2 = 13{,}4375_{10}$

Please note that the binary number is the same as in previous exampel, but in this exampel it's an integer with fraction.
(4 bits integer and 4 bits fraction.)

Compare with:
2019 is not equal to 20,19

Binary numbers, 2-complement form.

Example, binary number with 8 bits, integer (= 1 Byte):

1	**1**	**0**	**1**	**0**	**1**	**1**	**1**	Binary number
$-(2^7)$	**2^6**	**2^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**	Positional value, power of 2
-128	**64**	**32**	**16**	**8**	**4**	**2**	**1**	Positional value, in decimal

$1 \cdot -(2^7) + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 =$

$-1 \cdot 128 + 1 \cdot 64 + 1 \cdot 16 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = -41$

Dvs:

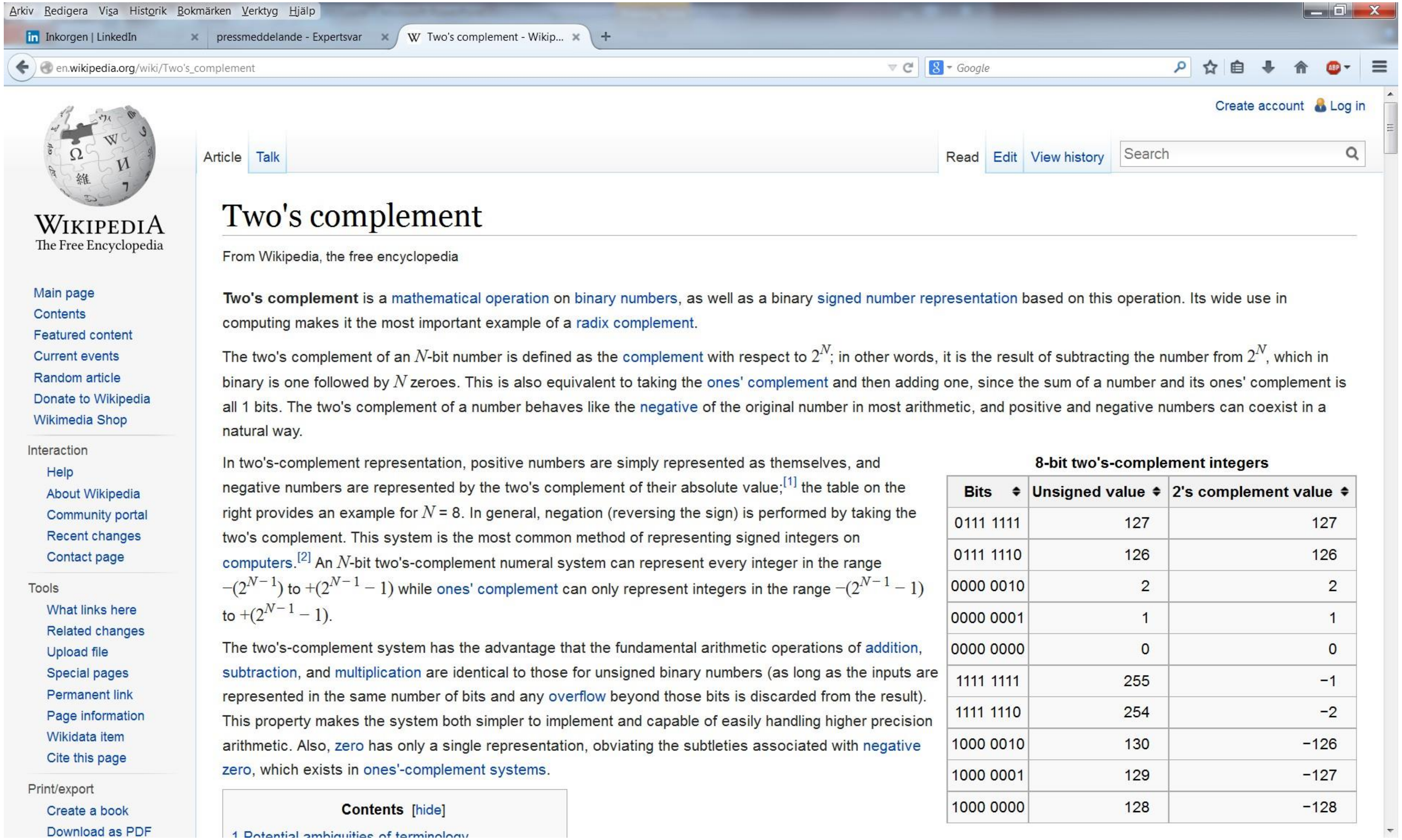
$1101\ 0111_2 = -41_{10}$

Radix 2 tells the base is 2, Binary number system

Radix 10 tells the base is 10, Decimal system

Two’s complement.

http://en.wikipedia.org/wiki/Two%27s_complement



Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp

Inkorgen | LinkedIn | pressmeddelande - Expertsvar | Two's complement - Wikip...

en.wikipedia.org/wiki/Two's_complement

Create account Log in

Article Talk Read Edit View history Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikimedia Shop

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF

Two's complement

From Wikipedia, the free encyclopedia

Two's complement is a mathematical operation on binary numbers, as well as a binary signed number representation based on this operation. Its wide use in computing makes it the most important example of a radix complement.

The two's complement of an N-bit number is defined as the complement with respect to 2^N; in other words, it is the result of subtracting the number from 2^N, which in binary is one followed by N zeroes. This is also equivalent to taking the ones' complement and then adding one, since the sum of a number and its ones' complement is all 1 bits. The two's complement of a number behaves like the negative of the original number in most arithmetic, and positive and negative numbers can coexist in a natural way.

In two's-complement representation, positive numbers are simply represented as themselves, and negative numbers are represented by the two's complement of their absolute value;[1] the table on the right provides an example for $N = 8$. In general, negation (reversing the sign) is performed by taking the two's complement. This system is the most common method of representing signed integers on computers.[2] An N-bit two's-complement numeral system can represent every integer in the range $-(2^{N-1})$ to $+(2^{N-1}-1)$ while ones' complement can only represent integers in the range $-(2^{N-1}-1)$ to $+(2^{N-1}-1)$.

The two's-complement system has the advantage that the fundamental arithmetic operations of addition, subtraction, and multiplication are identical to those for unsigned binary numbers (as long as the inputs are represented in the same number of bits and any overflow beyond those bits is discarded from the result). This property makes the system both simpler to implement and capable of easily handling higher precision arithmetic. Also, zero has only a single representation, obviating the subtleties associated with negative zero, which exists in ones'-complement systems.

8-bit two's-complement integers

Bits	Unsigned value	2's complement value
0111 1111	127	127
0111 1110	126	126
0000 0010	2	2
0000 0001	1	1
0000 0000	0	0
1111 1111	255	−1
1111 1110	254	−2
1000 0010	130	−126
1000 0001	129	−127
1000 0000	128	−128

Contents [hide]

Hexadecimal numbers.

In the hexadecimal positional system the base is 16. => 16 digits, 0-9, A, B, C, D, E, F.

Decimal:	Hexa-decimal:	Binary:
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
18	12	10010

$1A_{16}$ hexadecimal can be written as:

$1 \cdot 16^1 + A \cdot 16^0 =$
$= 1 \cdot 16^1 + 10 \cdot 16^0 =$
$= 1 \cdot 16 + 10 \cdot 1 =$
$= 16 + 10 = 26_{10}$

$1A9_{16}$ hexadecimal can be written as :

$1 \cdot 16^2 + A \cdot 16^1 + 9 \cdot 16^0 =$
$= 1 \cdot 16^2 + 10 \cdot 16^1 + 9 \cdot 16^0 =$
$= 1 \cdot 16 \cdot 16 + 10 \cdot 16 + 9 \cdot 1 =$
$= 1 \cdot 256 + 10 \cdot 16 + 9 \cdot 1 =$
$= 256 + 160 + 9 = 425_{10}$

Hexadecimal numbers.

In the hexadecimal positional system the base is 16. => 16 digits: 0-9, A, B, C, D, E, F.

Decimalt:	Hexa-decimalt:	Binärt:
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
18	12	10010

1	**1**	**0**	**1**	**0**	**1**	**1**	**1**
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
8	**4**	**2**	**1**	**8**	**4**	**2**	**1**

Groups of 4 bits from right.
$1101 = 8 + 4 + 0 + 1 = 13_{10} = D_{16}$

Groups of 4 bits from right.
$0111 = 0 + 4 + 2 + 1 = 7_{10} = 7_{16}$

Dvs = $1101\ 0111_2 = 215_{10} = D7_{16}$

The hexadecimal system.

In the hexadecimal positional system the base is 16.

2019 decimal can be written as:

$2019 = 2 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 9 \cdot 10^0 = 2000 + 0 + 10 + 9 = 2019$

$1101\ 0111_2$ binary can be written as:

$1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 215_{10}$

$D7_{16}$ hexadecimal can be written as:

$D \cdot 16^1 + 7 \cdot 16^0 = 13 \cdot 16 + 7 \cdot 1 = 208_{10} + 7 = 215_{10}$

Binary and hexadecimal numbers.

0001 1110 1101 0000 0010 0001

$0 * 2^{23} + 0 * 2^{22} + 0 * 2^{21} + 1 * 2^{20} + 1 * 2^{19} + 1 * 2^{18} + 1 * 2^{17} + 0 * 2^{16}$ +…..

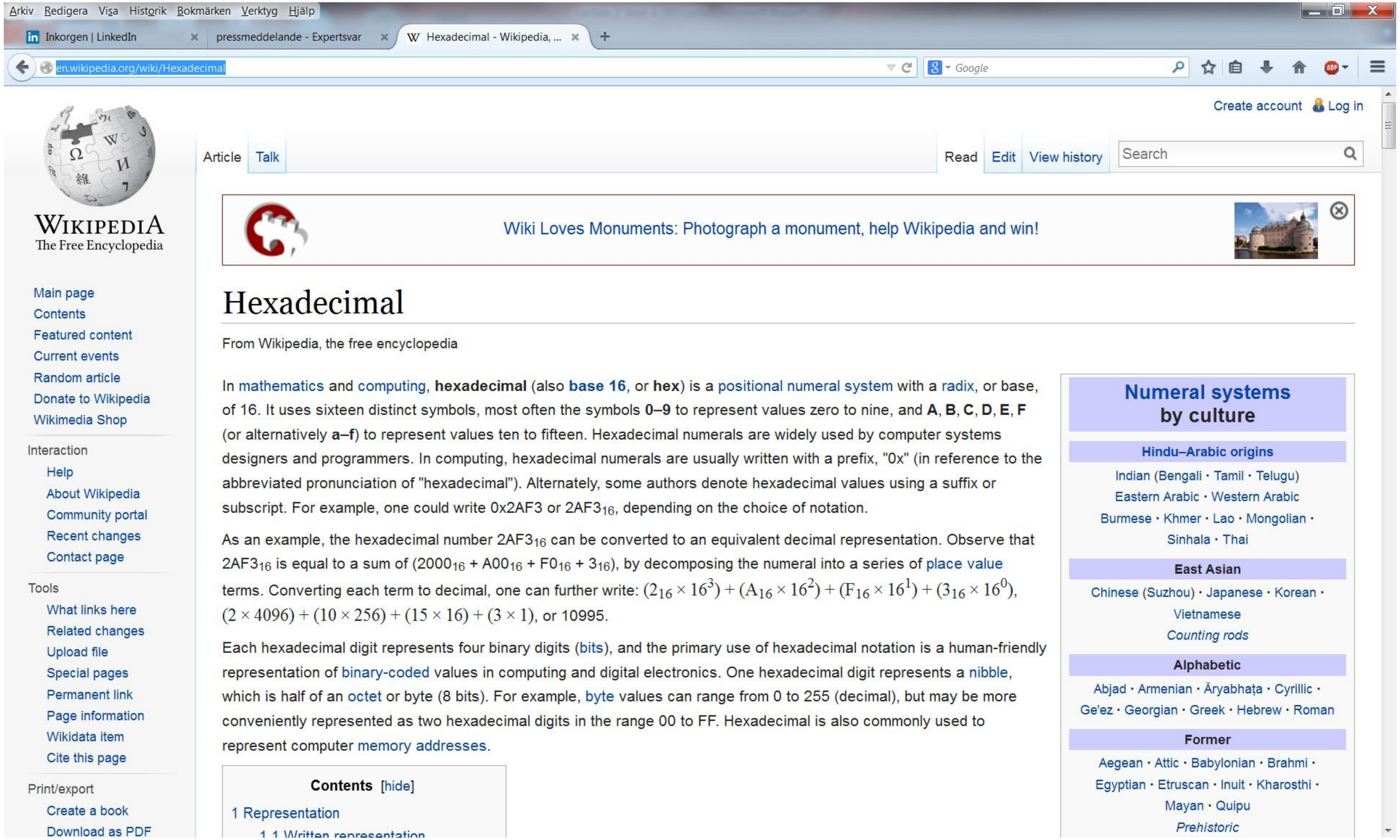
1 E D 0 2 1

$1 * 16^5 + 14 * 16^4 + 13 * 16^3 + 0 * 16^2 + 2 * 16^1 + 1 * 16^0 =$

$1 * 1\,048\,576 + 14 * 65\,536 + 13 * 4096 + 0 * 256 + 2 * 16 + 1 * 1 =$
$2\,019\,361_{10}$

Hexadecimal numbers.

http://en.wikipedia.org/wiki/Hexadecimal



MOV – Copy Register

Description:

This instruction makes a copy of one register into another. The source register Rr is left unchanged, while the destination register Rd is loaded with a copy of Rr.

Operation:

(i) $Rd \leftarrow Rr$

	Syntax:	Operands:	Program Counter:
(i)	MOV Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode:

0010	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formula:

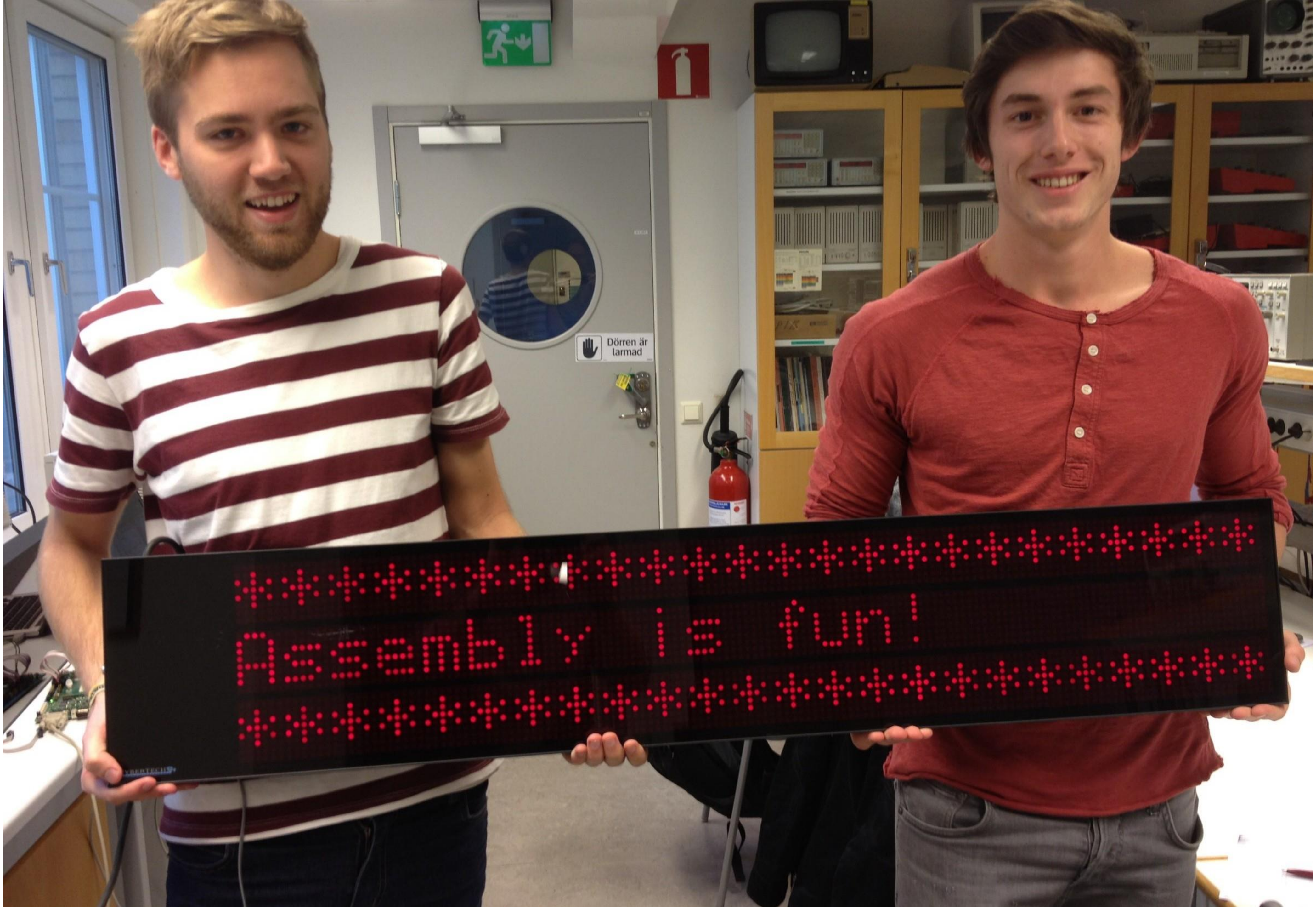
I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
         mov    r16,r0    ; Copy r0 to r16
         call   check     ; Call subroutine
         ...
check:   cpi    r16,$11   ; Compare r16 to $11
         ...
         ret              ; Return from subroutine
```

Words: 1 (2 bytes)

Cycles: 1


Dörren är
larmad
Assembly is fun!