

Mikroprocesory μP i mikrokontrolery μC - wykład 8

Adam Szmigielski

aszmigie@pjwstk.edu.pl

Maszyny liczące - rys historyczny

- nacięcia na drewnie, znaki na ścianach
- *pierwszy kalendarz* - Stonehenge (obecnie Salisbury, Anglia) skonstruowany ok. 2800 r. pne.
- *calculi* - kamyczki do liczenia u starożytnych Rzymian
- *abacus* - pierwsze liczydła (600-500 pne - Egipt lub Chiny)
- 650 r. - Hindusi odkrywają numeryczne zero - początek obliczeń pisanych.
- 1100 r. - pierwsza tabliczka mnożenia na piśmie
- 1612 - Szkot John Napier (1550-1617) odkrywa logarytmy i używa kropki dziesiętnej (wynalezionnej w Holandii)
- 1617 - narzędzie pomagające w mnożeniu - “kostki Napiera”
- 1622 - William Oughtred (1574-1660) tworzy *suwak logarytmiczny*.
- 1623 - Wilhelm Schickard (1592-1635) skonstruował czterodziałaniowy kalkulator-zegar.

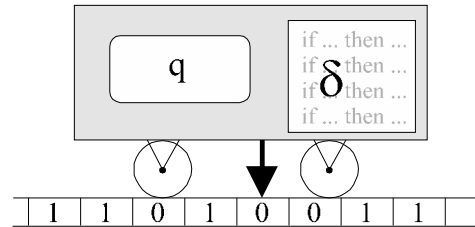
- 1642 - Blaise Pascal (1623-1662) tworzy "Pascalene" - 5-cio cyfrową *maszynę do dodawania*, uznaną za *pierwszą maszynę liczącą*.
- Gottfried Wilhelm von Leibniz (1646-1716) buduje czterodziałaniową maszynę liczącą
- 1822 - Charles Babbage (1792-1871) rozpoczął budowę maszyny do obliczeń nawigacyjnych.
- 1842 - Ada Augusta King (córka Lorda Byrona) pierwszą programistką (użyła maszyny Babbage'a)
- 1854 - George Boole opracowuje rachunek logiczny,
- 1903 - Nicola Tesla patentuje elektryczne bramki logiczne
- 1935-1938 - Konrad Zuse (1910- 1995) buduje **Z1** - *pierwszy komputer na przekaźnikach* (system dwójkowy).

Komputer współczesny

- 1937 - **Alan Turing (1912-1954)** rozwija teorię maszyny uniwersalnej (wykonującej algorytmy)
- 1941 - Zuse tworzy **Z3** z wykorzystaniem arytmetyki zmiennoprzecinkowej
- 1943 - *Colossus* - komputer deszyfrujący
- 1944 - Howard Aiken (1900-1973) i inżynierowie z IBM budują Harvard Mark
- 1945 - John von Neumann publikuje ideę “maszyny z Princeton”
- 1943-1946 *ENIAC* - pierwszy komputer na lampach (Uniwersytet Pensylwania)
- 1948 - *EDSAC* komputer oparty na idei von Neumanna (Cambridge)
- 1949 - *EDVAC* - komputer uniwersalny von Neumanna
- 1950 - *ACE* - komputer zbudowany według projektu Turinga
- 1951 - *UNIVAC* - pierwszy komercyjnie sprzedawany komputer

- 1954 - *IBM 704* - pierwszy komputer z systemem operacyjnym
- 1963 - *DEC PDP-5* - pierwszy minikomputer
- 1964 - komputery trzeciej generacji na obwodach scalonych
- 1971 - *Intel 4004* - pierwszy mikroprocesor
- 1972 - *Cray Research* - pierwsze superkomputery
- 1974 - *procesor Intel 8080*
- 1975 - komputer osobisty *Altair* - 256 bajtów pamięci
- 1981 - początek ery komputerów osobistych - pierwszy IBM PC

Algorytm i maszyna Turinga



Opis formalny - $\{Q, \Sigma, \delta, q_0, F\}$, gdzie:

- Q - zbiór *stanów maszyny*,
- Σ - *alfabet* (zbiór symboli) taśmy,
- δ - *funkcja przejścia*:

$$\delta : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{R, L, N\}$$

R, L, N odpowiadają kierunkowi przemieszczenia się czytelnika na taśmie.

- q_0 - *początkowy stan*,
- F - *zbiór końcowych stanów*.

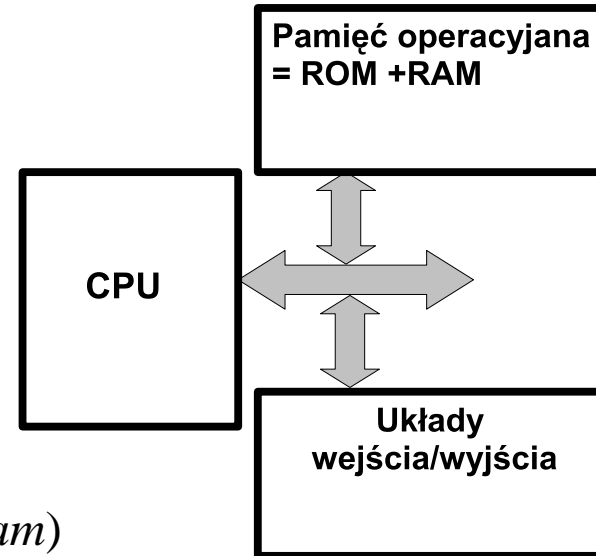
Komputer von Neumanna - 1945

Elementy składowe komputera von Neumanna:

- procesor z ALU
- pamięć komputera (zawierająca *dane i program*)
- urządzenia wejścia/wyjścia

Cechy komputera von Neumanna:

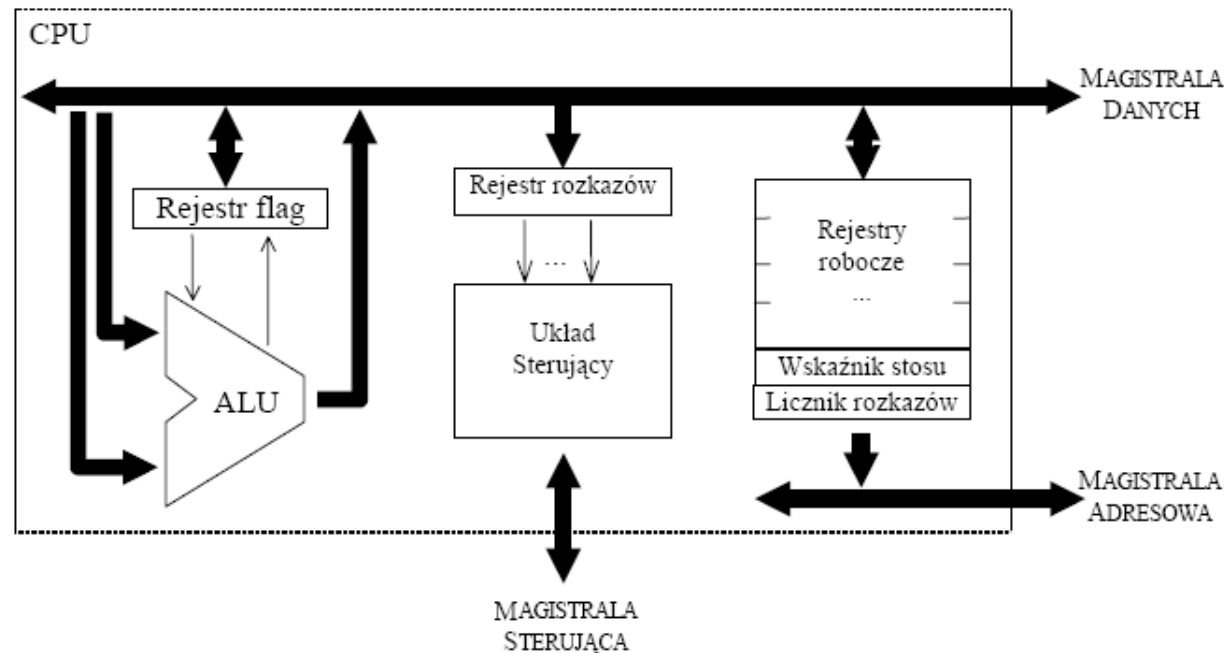
- skończona lista rozkazów,
- możliwość wprowadzenia programu i jego przechowywanie w pamięci (tak jak dane),
- sekwencyjne odczytywanie instrukcji z pamięci i ich wykonywanie.



Architektura harwardzka

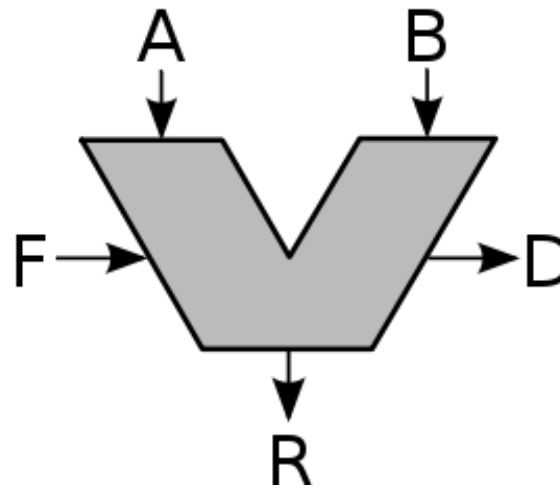
- *Pamięć danych programu* jest oddzielona od *pamięci rozkazów* (inaczej niż w architekturze von Neumanna).
- Prostsza, w stosunku do architektury von Neumanna, budowa ma większą szybkość działania - wykorzystuje się w procesorach sygnałowych oraz przy dostępie procesora do pamięci cache.
- *Architektura harwardzka* jest obecnie powszechnie stosowana w *mikrokomputerach jednoukładowych* (program w pamięci ROM (ang. Read Only Memory), dane w RAM (Random Access Memory)).

Architektura procesora vs organizacja



Rysunek 1: Architektura procesora - funkcjonalna

ALU i układ sterujący



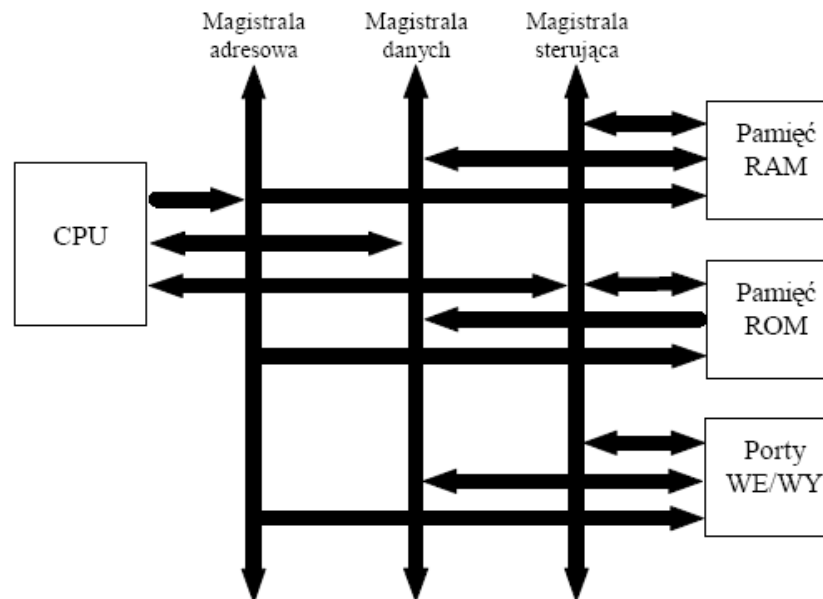
Rysunek 2: A i B - dane; R - wyjście; F - wybór operacji; D - status wyjścia

- *Jednostka Arytmetyczno-Logiczna* (ang. Arithmetic Logic Unit, ALU) układ kombinacyjny, wykonujący na danych w rejestrach operacje arytmetyczne (np. suma, różnica) oraz logiczne (np. OR, AND),
- *Układ sterujący* - dekoduje rozkazy i steruje jego wykonaniem.

Rejestry procesorze

- *Akumulator A, ACC* - rejestr bezpośrednio współpracujący z ALU
- *Wskaźnik stosu SP* - wskazuje koniec stosu (wyróżnionego obszaru pamięci)
- *Licznik rozkazów PC* - adres komórki pamięci programu z następnym rozkazem do wykonania
- *Rejestr rozkazów* - zawiera kod rozkazu wykonywanego rozkazu
- *Rejestr flag* - zawiera flagi (znaczniki bitowe) ustawiane w zależności od wyniku wykonanej operacji (np. nadmiar, zero, bit parzystości)
- *rejestry ogólnego przeznaczenia* - robocze

Magistrale systemu μ -procesorowego



- *magistrala adresowa* - przesyła adres (wybiera komórkę pamięci lub urządzenie we/wy),
- *magistrala danych* - przesyła dane między μP a pamięcią lub urządzenie we/wy),
- *magistrala systemowa* - zawiera sygnały sterujące.

Wielkość rejestru i słowo maszynowe

Słowo maszynowe jest ilość informacji, przetwarzanej w jednym rozkazie, tj.

- odpowiada wielkości rejestra (wyrażonej w bitach),
- zazwyczaj jest wielokrotnością bajta,
- odpowiada szerokości magistrali danych.

Cykl rozkazowy

Format rozkazu:

kod rozkazu	argumenty rozkazu
-------------	-------------------

Cykl rozkazu:

- *pobranie kodu rozkazu* - pobranie do rejestru rozkazu kodu rozkazu. Kody rozkazów przechowywane są w pamięci tak jak dane (architektura von Neumanna)
- *zdekodowanie rozkazu* - interpretacja wczytanego kodu rozkazu (zazwyczaj bajtu) jako polecenia z listy rozkazów procesora
- *wykonanie rozkazu* - wczytanie kolejnych argumentów rozkazu, w zależności od konkretnego rozkazu wykonanie ciągu operacji przez układ sterujący. Zapisanie wyniku w pamięci zewnętrznej lub rejestrze procesora

Cechy architektury CISC

CISC (ang. *Complex Instruction Set Computers*) – nazwa architektury mikroprocesorów o następujących cechach:

- Występowanie złożonych, specjalistycznych rozkazów (instrukcji) - wymagają od kilku do kilkunastu cykli maszynowych (zmienna liczba cykli),
- Szeroka gama trybów adresowania (skomplikowana konstrukcja dekodерów adresu),
- Stosunkowo długa listy rozkazów procesora.

Wady architektury CISC:

- zbyt długa lista rozkazów - część z nich jest rzadko używana,
- zbyt dużo czasu traci się na operacje przepisania z pamięci do rejestrów i odwrotnie,
- ogólnie mała efektywność w obliczeniach numerycznych.

Cechy architektury RISC

RISC (ang. *Reduced Instruction Set Computers*) - nazwa architektury mikroprocesorów o następujących cechach:

- Zredukowana liczba rozkazów do niezbędnego minimum
- Redukcja trybów adresowania, dzięki czemu kody rozkazów są prostsze, bardziej zunifikowane, (upraszcza dekodery rozkazów).
- Ograniczenie komunikacji pomiędzy pamięcią, a procesorem.
- *Przetwarzanie potokowe* (ang. *pipelining*) - równoległe wykonywanie rozkazów.

Obecnie popularne procesory z punktu widzenia programisty są widziane jako CISC, ale ich rdzeń jest RISC-owy. Rozkazy CISC są rozbijane na mikrorozkazy (ang. *microops*), które są następnie wykonywane przez RISC-owy blok wykonawczy.

Mikrokontrolery

Mikrokontroler - komputer zrealizowany w postaci pojedynczego układu scalonego, zawierającego jednostkę centralną (CPU), pamięć RAM, na ogół, pamięć programu oraz rozbudowane układy wejścia-wyjścia.

Określenie mikrokontroler pochodzi od głównego obszaru zastosowań, jakim jest sterowanie urządzeniami elektronicznymi.

Budowa mikrokontrolerów

Typowy mikrokontroler zawiera:

- Jednostkę obliczeniową (ALU) - przeważnie 8-bitową,
- Pamięć danych (RAM),
- Pamięć programu,
- Uniwersalne porty wejścia - część tych portów może pełnić alternatywne funkcje, wybierane programowo,
- Kontrolery transmisji szeregowej lub równoległej (UART, SPI, I2C, USB, CAN, itp.),
- Przetworniki analogowo-cyfrowe lub cyfrowo-analogowe,
- timery,
- Układ kontroli poprawnej pracy (watchdog)
- wewnętrzne czujniki wielkości nielektrycznych (np. temperatury)

Taktowanie mikrokontrolerów

Zegar systemowy mikrokontrolera może być taktowany:

- *zewnętrznym sygnałem taktującym* (rozwiązanie często stosowane w dużych układach wymagających synchronicznej współpracy wielu jednostek),
- *własnym generatorem*, wymagającym podłączenia zewnętrznych elementów ustalających częstotliwość taktowania (najczęściej jest to rezonator kwarcowy i dwa kondensatory),
- *wewnętrznym układem taktującym*, nie wymagającym podłączenia dodatkowych elementów

Zegary współczesnych mikrokontrolerów osiągają częstotliwości do kilkuset MHz, jednak w większości zastosowań taktowanie może być znacznie wolniejsze.

Języki programowania μP

- Języki wysokiego rzędu (np. VB, C, Java)
- Asembler

Sposoby programowania μC

Pamięci programu ROM można programować na trzy sposoby:

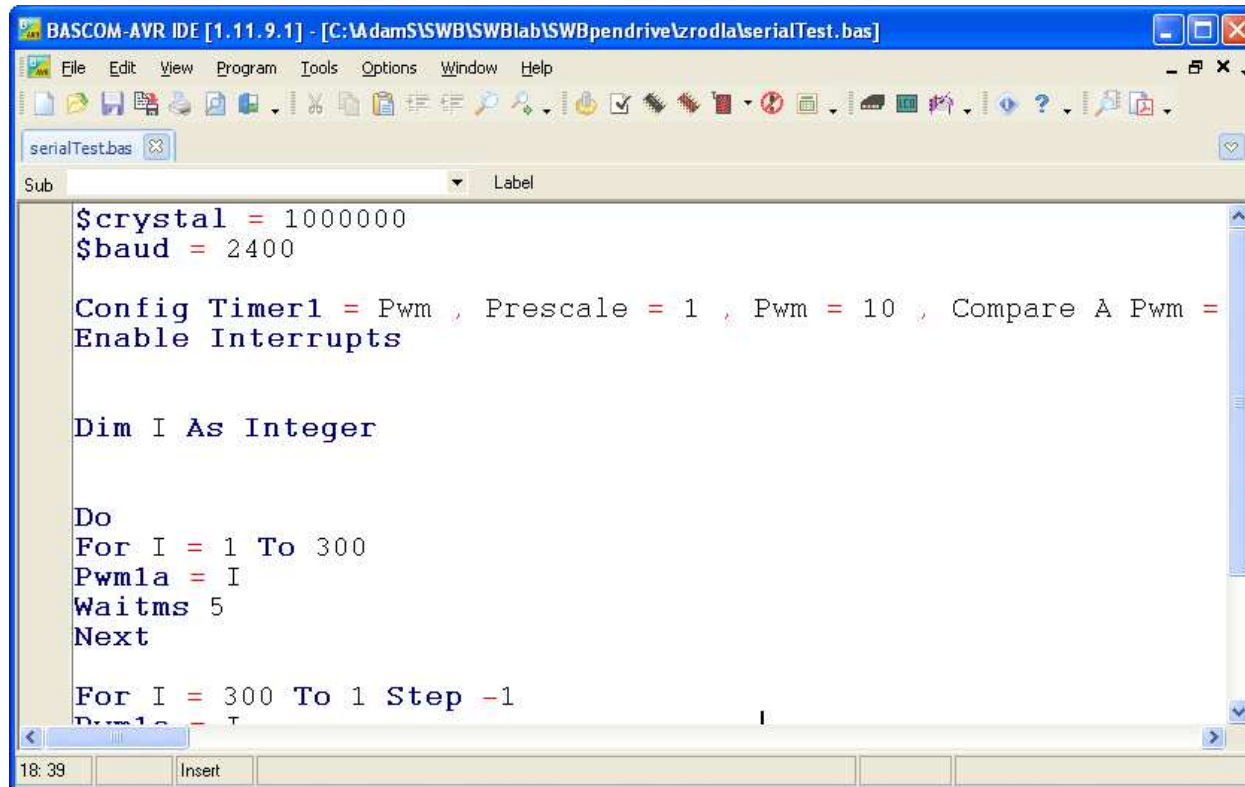
1. *High voltage Programming* czyli sposób programowania wprowadzony ponad 15lat temu do programowania pamięci EPROM za pomocą sygnałów 12V - wymaga programatora.
2. *ISP (In-System Programmable)* które nie wymaga wyjmowania pamięci z systemu w którym pracuje.
3. *Bootloader* - po resecie μC uruchamiany jest program znajdujący się w sekcji Bootloadera, który poprzez łącze (np. port szeregowy) łączy się z komputerem nadrzędnym, pobiera kod programu i umieszcza go w przeznaczony do tego obszarze pamięci ROM.

Przegląd obecnych mikrokontrolerów

Do najbardziej popularnych mikrokontrolerów należą:

1. Niekwestionowany standard dla rynku masowego narzuciła firma *Intel*, która wprowadziła na rynek mikrokontroler *8051*,
2. Bardzo popularne są również mikrokontrolery *AVR* firmy *Atmel* - w oparciu o nie będą prowadzona zajęcia laboratoryjne,
3. *PIC* firmy *Microchip Technology*,
4. inne.

Bascom-AVR



```
Sub
$crystal = 1000000
$baud = 2400

Config Timer1 = Pwm, Prescale = 1, Pwm = 10, Compare A Pwm =
Enable Interrupts

Dim I As Integer

Do
For I = 1 To 300
Pwm1a = I
Waitms 5
Next

For I = 300 To 1 Step -1
Pwm1a = I
```

- Wygląd głównego okna programu Bascom-AVR

Sprzęt obsługiwany przez Bascom-AVR

- Sprzęt zintegrowany w układzie scalonym
 - Timery (TIMER0 i TIMER1) i liczniki,
 - Rejestry wewnętrzne,
 - Port A i B,
 - Watchdog,
- obsługa zewnętrznych urządzeń
 - LCD
 - UART - możliwość emulator terminala
 - I2C
 - 1 WIRE protocol
 - SPI protocol w tym *In System Programming (ISP)*.

Język programowania używany w Bascom-AVR

- BASIC
- Assembler

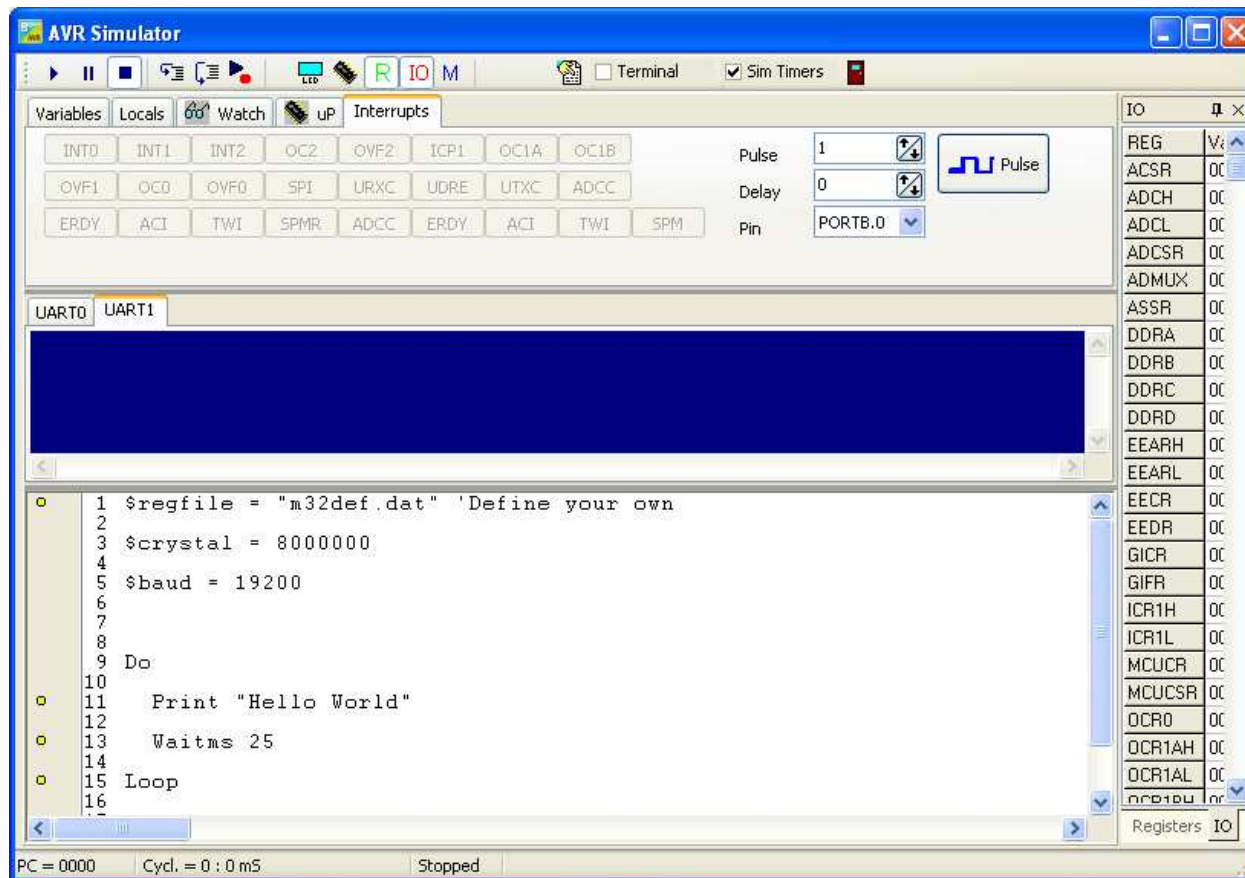
Basic - znaki i operatory

Character	Name
ENTER	Terminates input of a line
	Blank (or space)
'	Single quotation mark (apostrophe)
*	Asterisks (multiplication symbol)
+	Plus sign
,	Comma
-	Minus sign
.	Period (decimal point)
/	Slash (division symbol) will be handled as \
:	Colon
"	Double quotation mark
;	Semicolon
<	Less than
=	Equal sign (assignment symbol or relational operator)
>	Greater than
\	Backslash (integer/word division symbol)
^	Exponent

Operator	Relation Tested	Expression
=	Equality	X = Y
<>	Inequality	X <> Y
<	Less than	X < Y
>	Greater than	X > Y
<=	Less than or equal to	X <= Y
>=	Greater than or equal to	X >= Y

Operator	Meaning
NOT	Logical complement
AND	Conjunction
OR	Disjunction
XOR	Exclusive or

Symulator w Bascom-AVR



The screenshot displays the AVR Simulator interface. The main window is titled "AVR Simulator" and contains several panels:

- Top Panel:** Includes a toolbar with simulation controls (play, pause, stop, reset) and checkboxes for "Terminal" and "Sim Timers".
- Variables Panel:** Shows tabs for "Variables", "Locals", "Watch", and "uP". Below these are buttons for various hardware modules: INTD, INT1, INT2, OC2, OVF2, ICP1, OC1A, OC1B, OVF1, OC0, OVF0, SPI, URXC, UDRE, UTXC, ADCC, ERDY, ACI, TWI, SPMR, ADCC, ERDY, ACI, TWI, SPM.
- Pulse Generator:** A section for generating a pulse signal with fields for "Pulse" (set to 1), "Delay" (set to 0), and "Pin" (set to PORTB.0). A "Pulse" button and a waveform icon are also present.
- UART Panel:** Shows tabs for "UART0" and "UART1". The content area is currently blank.
- Code Editor:** Displays the following assembly code:

```
1 $regfile = "m32def.dat" 'Define your own
2
3 $crystal = 8000000
4
5 $baud = 19200
6
7
8
9 Do
10
11   Print "Hello World"
12
13   Waitms 25
14
15 Loop
16
```
- Registers Panel:** A list of registers on the right side of the window, including REG, ACSR, ADCH, ADCL, ADCSR, ADMUX, ASSR, DDRA, DDRB, DDRC, DDRD, EEARH, EEARL, EECR, EEDR, GICR, GIFR, ICR1H, ICR1L, MCUCR, MCUCSR, OCR0, OCR1AH, OCR1AL, and PORTB. The "Registers" tab is selected.
- Status Bar:** At the bottom, it shows "PC = 0000", "Cycl. = 0 : 0 mS", and "Stopped".

Zadania na ćwiczenia

1. Napisz program, który poprzez łącze szeregowo, odczytywać będzie wciśnięty klawisz oraz wyśle go na konsolę wraz jego kodem ASCII ^a,
2. Napisz funkcję, która będzie oczekiwać przez okres pięciu sekund na wciśnięcie klawisza. Jeśli w tym okresie klawisz zostanie wciśnięty powinna zwrócić jego kod ASCII, w przeciwnym przypadku 0. Funkcję należy wywoływać, wypisując na konsolę wciśnięty klawisz albo informację o nie wciśniętym klawiszu ^b,
3. Napisz program, wykorzystujący funkcję z poprzedniego punktu, który będzie klasyfikować wciśnięty klawisz jako: literę 'A' (małą lub dużą) lub cyfrę. Jeśli odczytany klawisz nie jest literą 'A' albo cyfrą należy wypisać odpowiednie komunikaty ^c.

^az wykorzystaniem funkcji *Print*, *Waitkey*, *Chr* oraz pętli *Loop*.

^bz wykorzystaniem funkcji *Waitms*, *Ischarwaiting* oraz pętli *For*. Funkcję należy zadeklarować z wykorzystaniem słowa kluczowego *Function* oraz użyć zmiennych lokalnych *Local*.

^cz wykorzystaniem funkcji *Select*.