1. Wstęp

Programowanie w asemblerze uważane jest za trudne zadanie. Jednak pisanie programów na najniższym poziomie daje możliwość poznania zasad działania mikroprocesora i pełnego wykorzystania jego zasobów. Asembler ma zarówno swoich zwolenników jak i przeciwników, jednak większość programistów zaleca przynajmniej podstawową znajomość asemblera. Celem niniejszej publikacji jest w sposób przystępny przedstawić studentom działanie mikrokontrolerów poprzez pisanie prostych programów w języku asembler.

O łatwości pisania programów w danym języku decyduje dostęp do bibliotek. Pisanie programów w językach wyższego poziomu np. w C++ czy Delphi bez gotowych bibliotek funkcji jest równie trudne i czasochłonne. Z tego powodu przygotowano gotowe biblioteki typowych procedur, a ćwiczenia laboratoryjne prezentowane w tej publikacji przygotowano w taki sposób, aby w największym stopniu ułatwić naukę programowania.

Temat programowania mikrokontrolerów AVR firmy Atmel w języku asembler jest dość często poruszany w literaturze [1, 2]. Dlatego w tej pozycji skupiono się na najważniejszych i podstawowych informacjach potrzebnych do rozpoczęcia programowania a szczegółowych informacji należy szukać w podanych pozycjach oraz nocie katalogowej mikrokontrolera ATmega16.

Skrypt przeznaczony jest głównie dla studentów kierunku automatyka i robotyka, a także elektronika, informatyka i kierunków pokrewnych.

Kody źródłowe procedur, opisanych w niniejszej pozycji są dostępne na stronie internetowej PWSZ w Raciborzu.

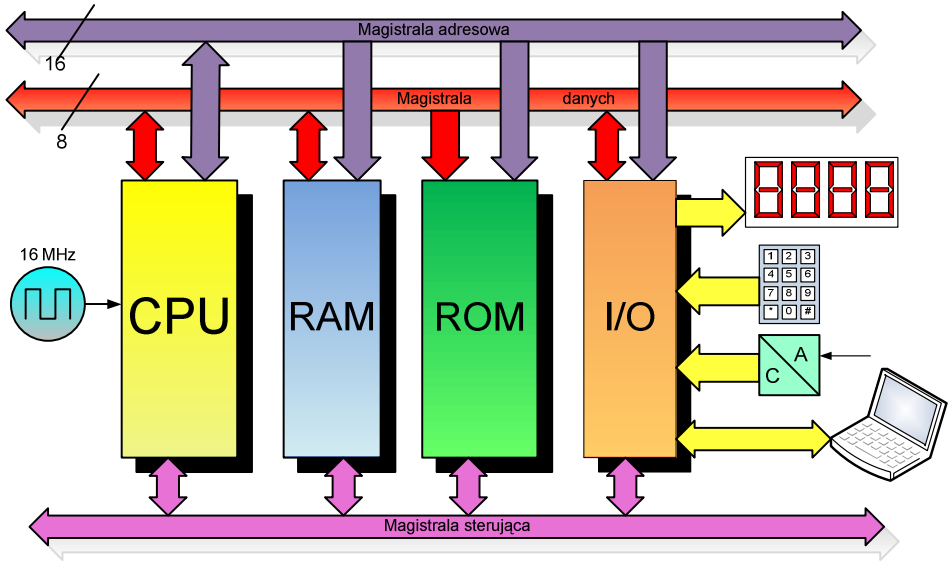
http://student.pwsz.raciborz.edu.pl/index.php/piotr-kalus-labolatorium-eitm

2. System mikroprocesorowy

2.1. Mikroprocesor i mikrokontroler

Mikroprocesor (w skrócie: µP lub uP) jest układem scalonym zawierającym jednostkę wykonującą po kolei rozkazy programu. Mikroprocesor, aby poprawnie pracować musi być połączony z kilkoma układami peryferyjnymi: pamięcią programu, pamięcią danych, układami wejścia/wyjścia, układami licznikowymi itd. Mikroprocesor z układami peryferyjnymi tworzy system mikroprocesorowy.

Mikrokontroler (w skrócie: µC lub uC) jest układem scalonym zawierającym w jednej obudowie wszystkie potrzebne do pracy układy peryferyjne.



Rys.1. Schemat blokowy systemu mikroprocesorowego

2.2. Mikrokontrolery AVR

Mikrokontrolery AVR produkuje firma Atmel. Obecnie są to jedne z najbardziej popularnych mikrokontrolerów. Do ich najważniejszych zalet można zaliczyć: dostępność na rynku, niską cenę, odporność na zakłócenia elektromagnetyczne oraz fakt, że są one nieustannie doskonalane. We wszystkich mikrokontrolerach rodziny AVR jest identyczny rdzeń, język asembler i sposób programowania. Układy peryferyjne w AVR-ach w większości są niemal identyczne. Różnice występują w wielkości pamięci programu i danych oraz w ilości układów peryferyjnych. Niniejsza publikacja skupia się głównie na mikrokontrolerze ATmega16, jednak z powodzeniem może służyć nauce programowania innych mikrokontrolerów rodziny AVR.

2.3. Pamięci mikrokontrolera AVR

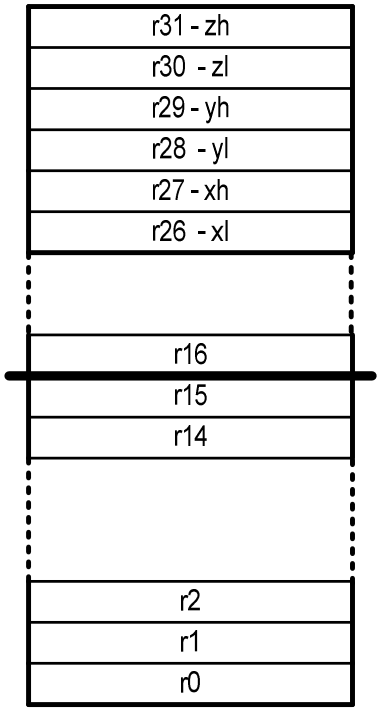
Mikrokontrolery AVR posiadają architekturę typu RISC, dlatego mają rozdzielone pamięci programu (FLASH) i danych (SRAM). Dodatkowo wyposażone są w pamięć danych EEPROM.

Pamięć **FLASH** zorganizowana jest jako pamięć 16-bitowa. Oznacza to, że każda komórka pamięci ma 16 bitów. W pamięci FLASH przechowywany jest program oraz dane startowe potrzebne w programie. Po odłączeniu zasilania zawartość tej pamięci nie jest tracona i po przywróceniu zasilania nie ma konieczności ponownie wgrywać programu.

Pamięć statyczna **RAM** (SRAM) wykorzystywana jest do przechowywania danych, które powstały w skutek działania programu. Po zaniku zasilania dane te są bezpowrotnie tracone. Pamięć SRAM zorganizowana jest w komórki 8-bitowe.

Pamięć **EEPROM** służy do przechowywania danych, które powstały w wyniku działania programu, lecz nie mogą one zostać utracone po zaniku napięcia zasilania. O ile pamięć SRAM nie ma limitu cykli zapisu/odczytu, to pamięć EEPROM nie może być kasowana i zapisywana w nieskończoność. Producent gwarantuje 100 000 cykli kasowania i zapisu. Należy więc zapisywać w niej dane, które nie będą zbyt często zmieniane.

2.4. Rejestry robocze mikrokontrolera



Rys.2. Rejestry robocze AVR

AVR posiada 32 rejestry robocze. Przez te rejestry dokonuje się większość operacji w mikrokontrolerze. Rejestry są 8-bitowe, co oznacza, że mogą przechowywać wartości od 0 do 255 (binarnie: od 00000000 do 11111111).

Rejestry robocze podzielone są na dwie części: rejestry „górne", czyli r16 do r31 i rejestry „dolne", czyli r0 do r15. Rejestry dolne mają mniejsze możliwości, m.in. nie można do nich bezpośrednio zapisać wartości. Należy to wykonać poprzez rejestry górne.

Sześć ostatnich rejestrów roboczych w niektórych operacjach mogą być wykorzystywane jako rejestry 16-bitowe X, Y i Z.

Rejestr X tworzą: r27 (bity starsze) i r26 (bity młodsze).

Rejestr Y tworzą: r29 (bity starsze) i r28 (bity młodsze).

Rejestr Z tworzą: r31 (bity starsze) i r30 (bity młodsze).

Dodatkowo para rejestrów r25 i r24 może być traktowana, jako 16-bitowy rejestr W, wykorzystywany do tworzenia pętli.

Ponieważ rejestry są 8-bitowe mogą przechowywać wartości od 0 do 255. Każdy bit rejestru ma swoją wagę w prezentacji liczby binarnej

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

Jeżeli w rejestrze jest zapisana wartość 10100011 to odpowiada to liczbie dziesiętnej 163, bo 128+32+2+1=163.

128	64	32	16	8	4	2	1	Wagi
1	0	1	0	0	0	1	1	=163

2.5. Układy peryferyjne i rejestry wejścia/wyjścia

Mikrokontroler AVR ATmega16 posiada następujące układy peryferyjne:

- Porty wejścia/wyjścia; pełnią funkcję wejść lub wyjść cyfrowych, połączonych z końcówkami mikrokontrolera.
- Liczniki – wykorzystywane m.in. do odmierzania odcinków czasowych, generowania impulsów prostokątnych lub jako generatory PWM.
- Interfejs szeregowy USART – do komunikacji z komputerem PC lub innymi mikrokontrolerami.
- Przetwornik A/C – 8-mio kanałowy, 10-bitowy przetwornik służy do pomiaru napięć.
- Interfejs szeregowy TWI – służy do komunikacji mikrokontrolera z innymi układani np. zewnętrznymi przetwornikami A/C lub C/A, pamięciami EEPROM, sterownikami itp.
- Interfejs szeregowy SPI – ma podobne zadania jak TWI, lecz jest szybszy i prostszy w obsłudze.
- Komparator analogowy – służy do detekcji zmian napięcia.
- Licznik WatchDog zabezpieczający mikrokontroler przed „zawieszeniem się programu”

Do obsługi układów peryferyjnych mikrokontroler wykorzystuje rejestry wejścia/wyjścia zwane czasami rejestrami specjalnymi. Służą one do konfiguracji pracy układów peryferyjnych oraz całego mikrokontrolera. Większość układów peryferyjnych obsługiwane jest przez trzy rodzaje rejestrów specjalnych: **rejestr danych**, **rejestr sterujący** i **rejestr statusu**. Przez rejestr danych wymieniane są dane między procesorem i układem peryferyjnym. Poprzez rejestr sterujący procesor konfiguruje pracę układu peryferyjnego, czyli włączenie/wyłączenie, tryb pracy, prędkość itp. Rejestr statusu służy do sprawdzenia stanu układu peryferyjnego, czyli np. czy układ zakończył zlecone operacje, czy pojawiły się błędy itp. Rejestry we/wy można odczytywać za pomocą rozkazu „in” oraz zapisywać za pomocą rozkazu „out”. Rejestry o adresach od 0 do 31 (dodatek B) mają możliwość dostępu do pojedynczych bitów poprzez rozkazy cbi, sbi, sbic i sbis.

Jednym z najważniejszych rejestrów wejścia/wyjścia jest rejestr SREG, nazywany rejestrem statusu procesora lub rejestrem flagowym. Podobny rejestr posiada każdy

mikroprocesor. Bity tego rejestru są uaktualniane po wykonaniu każdego rozkazu arytmetyczno-logicznego. Poniżej przedstawiono bity (flagi) tego rejestru.

128	64	32	16	8	4	2	1	
I	T	H	S	V	N	Z	C	SREG

I – flaga globalnego zezwolenia na przerwania. Gdy I=0 to przerwania są zablokowane. Gdy I=1 to przerwania są włączone. Bit I ustawia się rozkazem „sei", a kasuje rozkazem „cli".

Z – flaga zera, ustawiana lub kasowana po wykonaniu każdego rozkazu. Bit Z=1, gdy wynik ostatniego rozkazu jest równy 0. W pozostałych wypadkach Z=0.

C – flaga przeniesienia lub pożyczki. Bit C=1, gdy np. wynik dodawania w rejestrach przekroczył wartość 255 i wtedy pełni rolę 9-tego bitu wyniku o wadze 2^8, czyli 256. Bit C wraz z bitem Z często wykorzystywany jest przy skokach warunkowych.

T – służy do kopiowania pojedynczych bitów rejestrów roboczych

H – flaga przeniesienia połówkowego między trzecim a czwartym bitem, przy operacjach BCD

N – flaga wartości ujemnej; N=0 to liczba dodatnia; N=1 to liczba ujemna

V – flaga przepełnienia dla liczb w kodzie U2

S – flaga znaku ; $S = V \oplus N$

2.6. Rejestry sterujące portami wejścia/wyjścia

W najprostszym przypadku mikrokontroler ATmega16 można traktować jako programowalny układ cyfrowy, w którym mamy do dyspozycji 32 cyfrowe końcówki. Końcówki podzielone są na 4 tzw. porty: port A (PA), port B (PB), port C (PC) i port D (PD). Poszczególne końcówki portu A mają nazwy: PA7, PA6, PA5, PA4, PA3, PA2, PA1 i PA0. Analogiczne nazwy mają końcówki portów PB, PC i PD. Każda końcówka portu może pracować jako wejście lub wyjście cyfrowe. Jeżeli dana końcówka portu pracuje jako wyjście to może być na niej stan logiczny '0' i wtedy jest wewnętrznie zwarta do potencjału 0 woltów lub może być na niej stan logiczny '1' i wtedy jest wewnętrznie zwarta do potencjału 5 woltów. Jeżeli do takiej końcówki (np. PB0) podłączymy diodę LED jak na rysunku 12, to kiedy podamy stan logiczny '0', wtedy dioda świeci. Gdy podamy logiczną '1' to dioda zgaśnie. Podanie zera logicznego nazywamy skasowaniem bitu, a podanie jedynki logicznej nazywamy ustawianiem bitu.

W przypadku pracy danej końcówki portu jako wejście cyfrowe możemy odczytać stan logiczny tej końcówki. Jeżeli do końcówki (np. PA7) podłączymy przycisk jak na rysunku 12 to naciśniętemu przyciskowi odpowiada stan logiczny '0' podczas odczytu, a nienaciśniętemu przyciskowi odpowiada stan logiczny '1'.

Każdy port wejścia/wyjścia jest obsługiwany przez 3 rejestry wejścia/wyjścia.

Port A obsługują rejestry o nazwach: DDRA, PINA i PORTA.

Port B obsługują rejestry o nazwach: DDRB, PINB i PORTB.

Port C obsługują rejestry o nazwach: DDRC, PINC i PORTC.

Port D obsługują rejestry o nazwach: DDRD, PIND i PORTD.