Programowanie mikrokontrolerów

Asembler AVR, część 1

Marcin Engel Marcin Peczarski

Instytut Informatyki Uniwersytetu Warszawskiego

6 października 2012

Struktura programu w Asemblerze

- Program zaczyna się dyrektywą `.INCLUDE`, która włącza definicje niezbędne dla konkretnego modelu mikrokontrolera.
- Dla ATmega16 będzie to

```
.INCLUDE "m16def.inc"
```

- lub

```
.INCLUDE "C:\VMLAB\include\m16def.inc"
```

- lub

```
.INCLUDE "C:\PROGRA~1\vmlab\include\m16def.inc"
```

- Włączany plik zawiera dyrektywę:

```
.DEVICE ATmega16
```

- Dyrektywy `.INCLUDE` możemy również użyć w dowolnym miejscu programu do włączenia kodu umieszczonego w innym pliku.

Struktura programu w Asemblerze, cd.

- Program składa się z trzech segmentów odpowiadających poszczególnym przestrzeniom adresowym mikrokontrolera:

```
.DSEG   ; segment danych, SRAM

.ESEG   ; segment danych nieulotnych, EEPROM

.CSEG   ; segment kodu, FLASH
```

- Segmenty można dowolnie przeplatać.

Definiowanie stałych symbolicznych

- Stałe możemy definiować następująco

```
.EQU stała = wyrażenie
.SET stała = wyrażenie
```

- Dyrektywa `.EQU` działa jednorazowo.
- Dyrektywa `.SET` pozwala na wielokrotne przedefiniowywanie stałej.
- Przykłady:

```
.EQU SYS_FREQ = 16
.EQU BLINKER_HALF_PERIOD = 3 * SYS_FREQ
```

Alternatywne nazwy rejestrów

- Aby poprawić czytelność kodu, rejestrom można przypisywać nazwy dyrektywą `.DEF`.
- W ten sposób zdefiniowano rejestry `XL`, `XH`, `YL`, `YH`, `ZL` i `ZH`.
- Przykład:

```
.DEF AKUMULATOR = R16
LDI  AKUMULATOR, 7      ; wykona się LDI R16, 7
```

Wyrażenia stałe

Wyrażenia mogą zawierać:

- liczby całkowite
  - dziesiętne, np.: `11`, `-6`
  - szesnastkowe, np.: `0xB`, `0xFA`, `$B`, `$FA`
  - ósemkowe, np.: `013`, `0372`
  - dwójkowe, np.: `0b00001011`, `0b11111010`
- liczby zmiennoprzecinkowe: `-0.141592`
- kody ASCII: `'a'`
- stałe symboliczne
- etykiety

Wyrażenia stałe

Wyrażenia złożone buduje się z innych wyrażeń za pomocą:

- operatorów analogicznych jak w języku C
- funkcji Asemblera, z których dwie najważniejsze to:
  - młodszy bajt: `LOW()`
  - starszy bajt: `HIGH()`
- Przykłady:

```
1 << URSEL | 1 << UPM1 | 1 << UCSZ1 | 1 << UCSZ0

LOW(T1 - 1)
```

Wymuszanie wartości etykiet

- Wartości etykiet wymusza się dyrektywą `.ORG`.
- Przykład: ustalenie pierwszego adresu za wektorem przerwań

```
.CSEG
.ORG $2A
START:
```

- Przykład: ustalenie pierwszego adresu w pamięci danych

```
.DSEG
.ORG $60
ZMIENNE:
```

- Przykład: wymuszenie, aby etykieta `E7` w pamięci programu miała adres bajtowy podzielny przez 256 (adres słowa podzielny przez 128)

```
.CSEG
.ORG (PC + $7F) & $FF80
E7:
```

Zmienne w pamięci danych

- Miejsce rezerwujemy dyrektywą `.BYTE`.
- Przykład: deklaracja zmiennej jednobajtowej i tablicy

```
.DSEG
.ORG $60
INDEX:          .BYTE   1
TABLICA:        .BYTE   200
```

- Przykłads: załadowanie do `R16` wartości elementu tablicy o indeksie `INDEX`

```
.CSEG
        LDS     R16, INDEX
        LDI     XL, LOW(TABLICA)
        LDI     XH, HIGH(TABLICA)
        ADD     XL, R16
        LDI     R16, 0
        ADC     XH, R16
        LD      R16, X
```

Zmienne inicjowane w pamięciach nieulotnych

- Używamy dyrektyw:
  - `.DB` dla zmiennych 1-bajtowych,
  - `.DW` dla zmiennych 2-bajtowych,
  - `.DD` dla zmiennych 4-bajtowych,
  - `.DQ` dla zmiennych 8-bajtowych.
- Przykłady:

```
.ESEG
E1:     .DW 3456          ; zmienna dwubajtowa

.CSEG
E2:     .DB -27, "114"    ; zajęte 4 bajty
```

Operacje arytmetyczne

- Dodawanie i odejmowanie

```
ADD  Rd, Rr
ADC  Rd, Rr
SUB  Rd, Rr
SBC  Rd, Rr
```

- Przykład: dodanie wartości rejestru `R16` do 16-bitowego rejestru `X` (ze zniszczeniem wartości w rejestrze `R16`)

```
ADD  XL, R16
LDI  R16, 0
ADC  XH, R16
```

Operacje arytmetyczne

- Odejmowanie stałej z zakresu od -128 do 255, tylko na rejestrach `R16`, ..., `R31`

```
SUBI Rd, K
SBCI Rd, K
```

- Przykład: dodanie 5 do rejestru `R16`

```
SUBI R16, -5
```

- Przykład: dodanie 3000 do rejestru `Y`

```
SUBI YL, LOW(-3000)
SBCI YH, HIGH(-3000)
```

Operacje arytmetyczne

- Dodawanie do pary rejestrów i odejmowanie od pary rejestrów stałej z zakresu od 0 do 63, tylko na rejestrach `R25:R24`, ..., `R31:R30`

  ```
  ADIW Rd+1:Rd, K
  SBIW Rd+1:Rd, K
  ```

- Przykład: dodanie 1 do pary rejestrów `R25:R24`

  ```
  ADIW R25:R24, 1
  ```

- Przykład: odjęcie 1 od rejestru `Z`

  ```
  SBIW ZH:ZL, 1
  ```

Operacje arytmetyczne

- Dodawanie i odejmowanie jedynki (nie modyfikuje znaczników `C` i `H`)

  ```
  INC  Rd
  DEC  Rd
  ```

- Negacja arytmetyczna

  ```
  NEG  Rd
  ```

Operacje logiczne

- Bitowy iloczyn, alternatywa i alternatywa wykluczająca

```
AND  Rd, Rr
OR   Rd, Rr
EOR  Rd, Rr
```

- Negacja bitowa

```
COM  Rd
```

- Ustawienie znaczników `Z`, `N`, `V`, `S` zgodnie z zawartością rejestru

```
TST  Rd          ; inny zapis dla AND Rd, Rd
```

Operacje logiczne

- Bitowy iloczyn i alternatywa ze stałą, tylko na rejestrach `R16`, ..., `R31`

```
ANDI Rd, K
ORI  Rd, K
```

- Ustawienie lub wyzerowanie bitów w rejestrze

```
SBR  Rd, K      ; inny zapis dla ORI Rd, K
CBR  Rd, K      ; inny zapis dla ANDI Rd, ~K
```

- W powyższych rozkazach stała `K` jest maską bitową, a nie numerem bitu.
- Przykład: zerowanie dwóch najmłodszych bitów rejestru `R16`

```
CBR  R16, 0b00000011
```

Przesunięcia i rotacje bitów

- Logiczne przesunięcie w lewo o jeden bit

```
LSL  Rd          ; inny zapis dla ADD Rd, Rd
```

- Logiczne przesunięcie w prawo o jeden bit

```
LSR  Rd
```

- Rotacja w lewo o jeden bit przez znacznik `C`

```
ROL  Rd          ; inny zapis dla ADC Rd, Rd
```

- Rotacja w prawo o jeden bit przez znacznik `C`

```
ROR  Rd
```

- Arytmetyczne przesunięcie w prawo o jeden bit

```
ASR  Rd
```

Przesunięcia i rotacje bitów

- Przykład: pomnożenie zawartości rejestru `X` przez 4

```
LSL  XL
ROL  XH
LSL  XL
ROL  XH
```

- Zamiana półbajtów

```
SWAP Rd
```

Skoki bezwarunkowe

- Skok bezpośredni

  `JMP  k`

- Skok względny

  `RJMP  k`

- Skok pośredni do adresu z rejestru `Z`

  `IJMP`

- Skok bezpośredni umożliwia skoczenie pod dowolny adres w pamięci programu.
- Skok względny umożliwia skoczenie pod adres odległy od bieżącego w zakresie od -2047 do 2048 słów, ale wykonuje się szybciej i zajmuje mniej pamięci programu.

Operacje na rejestrach wejścia-wyjścia

- Odczyt

  ```
  IN   Rd, A
  ```

- Zapis

  ```
  OUT  A, Rr
  ```

- Przykład: odczyt rejestru stanu do rejestru `R16`

  ```
  IN   R16, SREG
  ```

- Wyzerowanie bitu, tylko rejestry o adresach od $00 do $1F

  ```
  CBI  A, b
  ```

- Ustawienie bitu, tylko rejestry o adresach od $00 do $1F

  ```
  SBI  A, b
  ```

- Przykład: ustawienie stanu wysokiego na wyprowadzeniu PA3 (musi być skonfigurowane jako wyjście)

  ```
  SBI  PORTA, PA3
  ```

Operacje na rejestrach we-wy

- Pominięcie następnej instrukcji, gdy bit wyzerowany

  `SBIC A, b`

- Pominięcie następnej instrukcji, gdy bit ustawiony

  `SBIS A, b`

- Przykład: aktywne oczekiwanie na zmianę stanu wyprowadzenia PA2 z wysokiego na niski (wyprowadzenie PA2 musi być skonfigurowane jako wejście)

```
CZEKAJ:
        SBIC PINA, PA2
        RJMP CZEKAJ
```

Wołanie procedur

- Bezpośrednie wołanie procedury

  `CALL k`
- Względne wołanie procedury

  `RCALL k`
- Pośrednie wołanie procedury spod adresu z rejestru `Z`

  `ICALL`
- Do terminów bezpośredni i pośredni odnoszą się te same uwagi co przy rozkazach skoków.
- Wołanie procedury odkłada na stos adres powrotu.
- Powrót z procedury

  `RET`
- Powrót z procedury (obsługi przerwania), włączenie przerwań

  `RETI`
- Powrót z procedury zdejmuje adres powrotu ze stosu.

Rozkazy warunkowe

- Pomiń następny rozkaz, gdy dwa rejestry mają tę samą wartość.

  `CPSE Rd, Rr`

- Pomiń następny rozkaz, gdy bit w rejestrze wyzerowany.

  `SBRC Rr, b`

- Pomiń następny rozkaz, gdy bit w rejestrze ustawiony.

  `SBRS Rr, b`

- Przykład: obliczenie wartości bezwzględnej

```
SBRC R16, 7
NEG  R16
```

Rozkazy porównujące wartości

- Porównaj wartość rejestrów, wykonaj odejmowanie `Rd - Rr`, ustaw znaczniki, ale nie zapisuj wyniku.

  ```
  CP   Rd, Rr
  ```

- Porównaj wartość rejestrów i znacznik przeniesienia `C`, wykonaj odejmowanie `Rd - Rr - C`, ustaw znaczniki, ale nie zapisuj wyniku.

  ```
  CPC  Rd, Rr
  ```

- Porównaj wartość rejestru ze stałą, wykonaj odejmowanie `Rd - K`, ustaw znaczniki, ale nie zapisuj wyniku, tylko na rejestrach `R16`, ..., `R31`.

  ```
  CPI  Rd, K
  ```

Rejestr stanu

- Znajduje się w pamięci we-wy pod adresem SREG ($3F).
- Jest uaktualniany po każdej operacji arytmetyczno-logicznej.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- bit I – włączone przerwania
- bit T – do przechowania dowolnego bitu (rozkazy BLD, BST)
- bit H – przeniesienie z młodszego półbajta do starszego
- bit V – nadmiar w arytmetyce U2
- bit N – najstarszy bit wyniku operacji (bit znaku w U2)
- bit S – znak rzeczywistego wyniku operacji w U2, $S = V \oplus N$
- bit Z – wynikiem operacji jest zero
- bit C – przeniesienie z najstarszego bitu

Rejestr stanu

odejmowanie	interpretacja w NKB	interpretacja w U2
10000010	130	−126
10000001	129	−127
00000001	1	1

`C = 0 N = 0 V = 0 S = 0`

odejmowanie	interpretacja w NKB	interpretacja w U2
10000010	130	−126
00000011	3	3
01111111	127	127

`C = 0 N = 0 V = 1 S = 1`

odejmowanie	interpretacja w NKB	interpretacja w U2
10000001	129	−127
00000001	1	1
10000000	128	−128

`C = 0 N = 1 V = 0 S = 1`

Rejestr stanu

odejmowanie	interpretacja w NKB	interpretacja w U2
10000001	129	−127
10000010	130	−126
11111111	255	−1

C = 1 N = 1 V = 0 S = 1

odejmowanie	interpretacja w NKB	interpretacja w U2
00000011	3	3
10000010	130	−126
10000001	129	−127

C = 1 N = 1 V = 1 S = 0

odejmowanie	interpretacja w NKB	interpretacja w U2
00000010	2	2
11111111	255	−1
00000011	3	3

C = 1 N = 0 V = 0 S = 0

Skoki warunkowe

- Skocz, gdy bit w rejestrze stanu ustawiony.

  `BRBS s, k`
- Skocz, gdy bit w rejestrze stanu wyzerowany.

  `BRBC s, k`
- Przykład: skocz do miejsca `ETYKIETA`, gdy bit `Z` (zero) wyzerowany.

  `BRBC 1, ETYKIETA`
- Żeby nie musieć pamiętać numerów bitów w rejestrze stanu, używamy następujących skrótów . . .

Skróty dla skoków warunkowych

rozkaz	warunek skoku	komentarz
BREQ	Z = 1	porównywane wartości równe wynik operacji zero
BRNE	Z = 0	porównywane wartości różne niezerowy wynik operacji
BRCS	C = 1	wystąpiło przeniesienie
BRLO		pierwszy argument mniejszy
BRCC	C = 0	nie wystąpiło przeniesienie
BRSH		pierwszy argument większy lub równy
BRVS	V = 1	wystąpiło przepełnienie (U2)
BRVC	V = 0	nie było przepełnienia (U2)
BRMI	N = 1	wynik operacji ujemny (U2)
BRPL	N = 0	wynik operacji nieujemny (U2)
BRLT	S = 1	pierwszy argument mniejszy (U2)
BRGE	S = 0	pierwszy argument większy lub równy (U2)

Skróty dla skoków warunkowych, cd.

rozkaz	warunek skoku	komentarz
`BRHS`	`H = 1`	wystąpiło przeniesienie z 3. bitu
`BRHC`	`H = 0`	nie było przeniesienia z 3. bitu
`BRTS`	`T = 1`	znacznik użytkownika ustawiony
`BRTC`	`T = 0`	znacznik użytkownika wyzerowany
`BRIE`	`I = 1`	przerwania włączone
`BRID`	`I = 0`	przerwania wyłączone

Skoki warunkowe, cd.

- Przykład: skok, gdy wartość w rejstrze `R18` jest równa 0 lub większa od 127

```
CPI  R18, 1
BRLT ETYKIETA
```

Operacje na znacznikach

- Skopiuj bit z rejestru do znacznika użytkownika `T`.

  ```
  BST  Rr, b
  ```

- Skopiuj bit ze znacznika użytkownika `T` do rejestru.

  ```
  BLD  Rd, b
  ```

- Ustaw znacznik.

  ```
  BSET s
  ```

- Wyzeruj znacznik.

  ```
  BCLR s
  ```

- Żeby nie musieć pamiętać numerów bitów w rejestrze stanu, używamy następujących skrótów . . .

Skróty operacji na znacznikach

rozkaz	operacja	rozkaz	operacja
SEC	C := 1	CLC	C := 0
SEZ	Z := 1	CLZ	Z := 0
SEN	N := 1	CLN	N := 0
SEV	V := 1	CLV	V := 0
SES	S := 1	CLS	S := 0
SEH	H := 1	CLH	H := 0
SET	T := 1	CLT	T := 0
SEI	I := 1	CLI	I := 0