DSM-51 v2
Dydaktyczny System Mikroprocesorowy

DSM51ASS (Asembler 8051) wersja 1.01

Producent:

MicroMade
Systemy Mikroprocesorowe

Siedziba Firmy:
ul. Sikorskiego 33
64-920 PIŁA
tel/fax: (67)13-24-14

Biuro Konstrukcyjne:
ul. Fiszera 14
80-231 GDAŃSK
tel/fax: (58)47-15-35

Program DSM51ASS jest asemblerem mikrokontrolera 8051 przeznaczonym do stosowania przy nauce programowania tego mikrokontrolera z wykorzystaniem Dydaktycznego Systemu Mikroprocesorowego DSM-51.

Program DSM51ASS jest dostarczany łącznie z systemem DSM-51 oraz rozprowadzany na dyskietce zawartej w książce "Podstawy programowania mikrokontrolera 8051".

Opis programu:

Podstawowe cechy asemblera DSM51ASS:

- jest makroasemblerem jednoprzebiegowym,
- komunikaty wypisuje w języku polskim (DOS strona kodowa 852),
- pozwala asemblować tylko pojedyńczy plik wejściowy

(nie ma fazy linkowania),
- asemblowany program może liczyć maksymalnie około
  20,000 linii listingu (zależnie od dostępnej pamięci),
- zezwala na stosowanie rozbudowanych wyrażeń arytmetycznych
  o postaci zbliżonej do wyrażeń języka C,
- wszystkie wyrażenia arytmetyczne liczy na liczbach 32 bitowych
  ze znakiem,
- wartości wszystkich symboli pamięta w postaci liczb 16 bitowych
  bez znaku.

Wywołanie programu:

```
DSM51ASS <nazwa>
```

gdzie <nazwa> jest nazwą pliku zawierającego kod źródłowy programu. Jeśli w nazwie nie zawarto rozszerzenia to domyślnie przyjmowane jest rozszrzenie .asm.

W wyniku działania programu powstają następujące zbiory:

<nazwa>.hex - zbiór zawierający kod wynikowy w formacie Intel HEX.
<nazwa>.lst - listing programu

Format linii programu:

Typowa linia programu w asemblerze wygląda następująco:

[<etykieta>] [<rozkaz>] [<operandy>] [;<komentarz>]

Znaczenie poszczególnych pól linii programu jest następujące:

<etykieta> - symbol umieszczony na samym początku linii (pierwszy znak etykiety musi być pierwszym znakiem w linii). Etykieta musi zaczynać się od litery lub znaku podkreślenia '_', i może zawierać dowolną kombinację liter, cyfr i podkreśleń. Jeśli etykieta jest zakończona dwukropkiem to nadawana jest jej wartość określająca jej pozycję w kodzie źródłowym (adres rozkazu z tej linii programu). Etykiety (symbole) stosowane z dyrektywami nadającymi im wartość nie są zakończone dwukropkiem.

<rozkaz> - mnemonik kodu maszynowego procesora, dyrektywa asemblera lub makro.

<operandy> - informacje wymagane przez mnemonik, dyrektywę asemblera lub makro. Poszczególne operandy są oddzielane przecinkami.

<komentarz> - wszystkie znaki występujące po średniku są traktowane jako komentarz i ignorowane przez asembler.

Poszczególne pola linii programu muszą być oddzielone między sobą co najmniej jednym znakiem spacji (lub tabulacji).
W programie mogą występować puste linie lub linie zawierające wyłącznie komentarz.

Wyrażenia arytmetyczne.

Wyrażenia arytmetyczne służą do określenia wartości parametrów

wymagających podania wartości liczbowej.
Wyrażenia składają się ze stałych liczbowych i symboli (etykiety, nazwy stałych lub zmiennych) połączonych operatorami arytmetycznymi.

Składnia wyrażeń asemblera DSM51ASS została zaczerpnięta z języka C. Wprowadzono kilka dodatkowych operatorów stosowanych powszechnie w asemblerach oraz zmieniono priorytet operacji bitowych - są one wykonywane przed operacjami porównań.
Operatory porównań dają wartość 1 jeżeli warunek jest prawdziwy oraz 0 jeżeli jest fałszywy.
Operatory logiczne (!,&&,||) każdą wartość różną od zera traktują jako prawdę, a wartość 0 jako fałsz. Jako wynik operacji logicznych również uzyskujemy wartość 1 lub 0.

W asemblerze DSM51ASS wszystkie obliczenia wykonywane są na liczbach 32 bitowych ze znakiem. Oznacza to, że wartość wyrażenia jest wyliczona prawidłowo dopóki wyniki pośrednie mieszczą się w zakresie -2,147,483,648 do 2,147,483,647. Przekroczenie tego zakresu w czasie obliczeń nie jest sygnalizowane.

Wartości symboliczne używane w programie są przechowywane jako liczby 16 bitowe bez znaku.

Asembler kontroluje wartość i typ wyrażenia. Jeżeli wartość wyrażenia arytmetycznego wykracza poza zakres dopuszczony dla aktualnego parametru lub typ wyrażenia jest nieodpowiedni to generowany jest błąd.

Stałe liczbowe.

Stała liczbowa musi zaczynać się od cyfry.

DSM51ASS akceptuje następujące typy stałych liczbowych:

Typ	Składnia	Przykład
Dziesiętny	<cyfry>	125
Szesnastkowy	<cyfra><cyfry szesnastkowe>H	0FFFFH
Ósemkowy	<cyfry ósemkowe>O	7777O
Binarny	<cyfry binarne>B	10101B
Znakowy	'<znak>'	'A'

Symbole.

Symbole reprezentowane są przez ciąg znaków zaczynający się od litery lub znaku podkreślenia '_' i składający się z dowolnej sekwencji liter,cyfr i podkreśleń.
Asembler rozpoznaje pierwsze 32 znaki symbolu.

W asemblerze DSM51ASS zdefiniowano standardowe symbole reprezentujące poszczególne rejestry i bity procesora 8051.
Poza tym asembler rozpoznaje symbole określające adresy urządzeń systemu DSM-51 i zawartych w jego pamięci EPROM podprogramów standardowych.

Listę tych symboli umieszczono w rozdziale 'Predefiniowane symbole' w końcowej części tego pliku.

Operatory arytmetyczne.

Operatory w/g priorytetu ich wykonywania:

```
(),              Nawiasy
!, ~, +, -, <, >       Modyfikacje wartości
.0, .1, .2, .3, .4, .5, .6, .7    Selekcja bitów
*, /, %                     Mnożenie, dzielenie
+, -                   Dodawanie, odejmowanie
<<, >>                      Przesunięcia bitowe
&                Bitowe AND
^                Bitowe XOR
|                Bitowe OR
<, <=, >, >=, =, ==, !=           Porównania
&&               Logiczne AND
||               Logiczne OR
```

Znaczenie poszczególnych operatorów:

() - Nawiasy określają kolejność wykonywania działań. Nie ma ograniczenia liczby zastosowanych zagłębionych nawiasów.

Operatory modyfikujące wartość nastepującego po nich operandu.

! - Negacja logiczna. Zmienia wartość różną od 0 na 0, a wartość równą 0 na 1.
~ - Negacja bitowa. Zmienia wszystkie 32 bity w operandzie na odwrotne.
+ - Nie zmienia wartości.
- - Zmienia znak operandu na przeciwny.
< - Najmłodszy bajt operandu (wyrażenie '< operand' jest równoważne wyrażeniu '(operand & 0FFH)' lub '(operand % 256)').
> - Strsze 3 bajty opernadu (wyrażenie '> operand' jest równoważne wyrażeniu '(operand >> 8)' lub '(operand / 256)').

Operatory selekcji bitów w bitowo adresowalnych rejestrach.

.n - Występując po adresie (nazwie) rejestru oblicza adres wskazanego bitu tego rejestru. Jeżeli dany adres nie jest adresem bitowo adresowalnego rejestru sygnalizowany jest błąd. 'n' jest cyfrą z zakresu 0..7.

Operatory mnożenia i dzielenia.

* - Mnożenie.
/ - Dzielenie.
% - Dzielenie 'modulo' (reszta z dzielenia).

Operatory dodawania i odejmowania.

+ - Dodawanie.
- - Odejmowanie.

Operatory przesunięć bitowych.

<< - Przesunięcie w lewo. Operand występujący z lewej strony operatora jest przesuwany w lewo o liczbę bitów określoną przez operand występujący z prawej strony. Na zwalniane bity wchodzą zera.
>> - Przesunięcie w prawo. Operand występujący z lewej strony operatora jest przesuwany w prawo o liczbę bitów określoną przez operand występujący z prawej strony. Na zwalniane bity wchodzą zera.

Operatory bitowe.

& - AND między bitami operandów. Odpowiedni bit w wyniku ma wartość '1' tylko wtedy, gdy odpowiadające mu bity w obu operandach mają wartość '1'. W pozostałych przypadkach ma wartość '0'. Operacja ta jest wykonywana dla wszystkich 32 bitów.

Operand1	Operand2	Wynik
0	0	0
0	1	0
1	0	0
1	1	1

^ - XOR między bitami operandów. Odpowiedni bit w wyniku ma wartość '1' tylko wtedy, gdy jeden z odpowiadających mu bitów w operandach ma wartość '1', a drugi '0'. W pozostałych przypadkach ma wartość '0'. Operacja ta jest wykonywana dla wszystkich 32 bitów.

Operand1	Operand2	Wynik
0	0	0
0	1	1
1	0	1
1	1	0

| - OR między bitami operandów. Odpowiedni bit w wyniku ma wartość '1', gdy co najmniej jeden z odpowiadających mu bitów w operandach ma wartość '1'. W przeciwnym przypadku ma wartość '0'. Operacja ta jest wykonywana dla wszystkich 32 bitów.

Operand1	Operand2	Wynik
0	0	0
0	1	1
1	0	1
1	1	1

Operatory porównań.

< - Mniejszy. Prawda (=1) gdy wartość lewego operandu jest mniejsza od wartości prawego operandu.
<= - Mniejszy lub równy. Prawda (=1) gdy wartość lewego operandu jest mniejsza od lub równa wartości prawego operandu.
> - Większy. Prawda (=1) gdy wartość lewego operandu jest większa od wartości prawego operandu.
>= - Większy lub równy. Prawda (=1) gdy wartość lewego operandu jest większa od lub równa wartości prawego operandu.
= - Równy. Prawda (=1) gdy wartość lewego operandu jest równa wartości prawego operandu.
== - Równy. Prawda (=1) gdy wartość lewego operandu jest równa wartości prawego operandu.
!= - Różny. Prawda (=1) gdy wartość lewego operandu jest różna od wartości prawego operandu.

Operatory logiczne.

&& - AND logiczne. Prawda (=1) gdy oba operandy mają wartości

różne od zera.

|| - OR logiczne. Prawda (=1) gdy co njmniej jeden z operandów ma wartość różną od zera.

Dyrektywy asemblera.

W poszczególnych liniach programu oprócz mnemoników oznaczających poszczególne rozkazy procesora mogą wystąpić dyrektywy asemblera. Umożliwiają one wstawianie danych w treść programu, przypisywanie wartości symbolom, sterowanie przebiegiem asemblacji i budowanie makr (zestawów poleceń wywoływanych pojedyńczą nazwą).

Asembler DSM51ASS akceptuje następujące dyrektywy:

Dyrektywy danych:

- DB - wstawienie w kod wartości numerycznych i tekstowych,
- DW - wstawienie w kod dwubajtowych wartości numerycznych,
- EQU - definiowanie stałej,
- BIT - definiowanie stałej typu bit,
- REG - definiowanie stałej typu rejestr,
- SET - definiowanie zmiennej.

Dyrektywy sterujące:

- IF - początek bloku warunkowej asemblacji,
- ELSE - początek alternatywnego bloku warunkowej asemblacji,
- ENDIF - koniec bloku warunkowej asemblacji,
- ORG - ustawienie adresu dla następnego bloku kodu,
- END - koniec programu.

Dyrektywy makrodefinicji:

- MACRO - początek definicji makra,
- ENDM (MACEND) - koniec definicji makra.

Dyrektywy danych.

DB - wstawienie w kod wartości numerycznych i tekstowych

Składnia:

[<etykieta>] DB <parametry>

Wpisuje w treść programu wartości parametrów. Poszczególne parametry oddzielane są przecinkami. Parametry mogą być wyrażeniami arytmetycznymi lub ciągami znaków ujętymi w znaki ' lub ".
Wartości kolejnych parametrów będących wyrażeniami arytmetycznymi są wpisywane w kolejne bajty treści programu. Parametry będące ciągami znaków są wpisywane w całości do treści programu. W ciągu znaków ujętym w znaki " może wystąpić znak ' i odwrotnie.

DW - wstawienie w kod dwubajtowych wartości numerycznych

Składnia:

[<etykieta>]   DW <parametry>

Wpisuje w treść programu wartości parametrów. Poszczególne
parametry oddzielane są przecinkami. Parametry są wyraże-
niami arytmetycznymi.
Wartość każdego parametru jest wpisywana w dwa kolejne
bajty treści programu (najpierw starszy).

EQU - definiowanie stałej.

Składnia:

<symbol> EQU   <wyrażenie>

Symbolowi <symbol> przypisywana jest wartość wyrażenia.
Typ symbolu ustalany jest na podstawie wyrażenia.
Każda wartość zdefiniowana dyrektywą EQU jest stałą i nie
może być zmieniana w trakcie asemblacji.

BIT - definiowanie stałej typu bit.

Składnia:

<symbol> BIT   <wyrażenie>

Symbolowi <symbol> przypisywana jest wartość wyrażenia.
Kontroluje typ wyrażenia. Zdefiniowany symbol może być
używany wyłącznie jako adres bitu. Wartość symbolu nie
może być zmieniana w trakcie asemblacji.

REG - definiowanie stałej typu rejestr.

Składnia:

<symbol> REG   <wyrażenie>

Symbolowi <symbol> przypisywana jest wartość wyrażenia.
Kontroluje typ wyrażenia. Zdefiniowany symbol może być
używany wyłącznie jako adres rejestru wewnętrznego
mikrokontrolera lub komórki wewnętrznej pamięci RAM.
Wartość symbolu nie może być zmieniana w trakcie asemblacji.

SET - definiowanie zmiennej.

Składnia:

<symbol> SET   <wyrażenie>

Symbolowi <symbol> przypisywana jest wartość wyrażenia.
Typ symbolu ustalany jest na podstawie wyrażenia.
Wartości zdefiniowane dyrektywą SET mogą być dowolnie wiele
razy modyfikowane przez ponowne użycie dyrektywy SET.
Zmiana typu symbolu w trakcie kolejnego przypisania powoduje
wygenerowanie ostrzeżenia.

Dyrektywy sterujące.

IF - początek bloku warunkowej asemblacji.

Składnia:

    IF <wyrażenie>

Jeżeli wartość wyrażenia jest różna od '0' (prawda) to
kod występujący za tą dyrektywą jest asemblowany. Jeżeli
wartość wyrażenia jest równa '0' (fałsz) i istnieje dyrektywa
ELSE to kod występujący po ELSE jest asemblowany.
Dyrektywa ENDIF zamyka blok kodu asemblowanego warunkowo.
Dopuszczalne jest zagłębianie dyrektyw IF do 16 poziomów.

ELSE - początek alternatywnego bloku warunkowej asemblacji.

Składnia:

    ELSE

Używana w połączeniu z dyrektywą IF. Jeśli wyrażenie testowane
w dyrektywie IF ma wartość '0' to alternatywny kod zaznaczony
przez ELSE jest asemblowany.

ENDIF - koniec bloku warunkowej asemblacji.

Składnia:

    ENDIF

Zakończenie bloku warunkowej asemblacji rozpoczętego dyrektywą IF.

ORG - ustawienie adresu dla następnego bloku kodu.

Składnia:

    ORG <wyrażenie>

Ustawienie adresu dla następującego po tej dyrektywie bloku kodu.
Adres dla następnej instrukcji procesora jest ustalany poprzez
wyliczenie wartości wyrażenia. Możliwe jest jedynie zwiększanie
aktualnego adresu kodu. Próba zmniejszenia adresu jest sygnalizo-
wana jako błąd.
Standardowo kod programu jest umieszczany rozpoczynając od adresu 0.

END - koniec programu.

Składnia:

    END

Zaznaczenie końca programu. Linie występujące w pliku źródłowym
po tej dyrektywie nie są asemblowane.
Użycie tej dyrektywy w programie nie jest konieczne. Przy jej braku
końcem programu jest koniec pliku.

Dyrektywy makrodefinicji.

MACRO - początek definicji makra.

Składnia:

<Nazwa> MACRO <parametry>

Makro to zestaw instrukcji asemblera.
Ciąg instrukcji występujący po linii zawierającej dyrektywę MACRO, aż do najbliższej dyrektywy ENDM, tworzy makro o nazwie <Nazwa>.
Po zdefiniowaniu cały taki zestaw może być włączony w kod źródłowy programu poprzez wywołanie makra.
W treści makr mogą występować bez ograniczeń wywołania innych makr, ale nie może wystąpić definicja innego makra. Próba wywołania przez makro samego siebie (lub innego zapętlenia wywoływań prowadzącego do nieskończonego rozwijania makr) jest wykrywana i sygnalizowana jako błąd.
Parametry to oddzielone przecinkami symbole (parametry formalne makra), które mogą być wykorzystywane w treści makra.

Makro jest wywoływane poprzez umieszczenie jego nazwy w polu rozkazu danej linii programu. Przy wywołaniu podawane są parametry aktualne makra.
W czasie wstawiania makra w kod programu asembler zastępuje wszystkie parametry formalne parametrami aktualnymi.

Asembler nie umożliwia tworzenia etykiet lokalnych w makrach. Jeśli występuje taka potrzeba to można podawać etykietę (lub jej fragment) jako jeden z parametrów makra. Jest to możliwe, gdyż zastępowanie parametrów formalnych parametrami aktualnymi odbywa się na drodze podmieniania tekstów przed asemblacją linii programu. Teksty te są podmieniane niezależnie od tego, czy występują w etykiecie, nazwie mnemonika, czy w treści operandu.

ENDM (MACEND) - koniec definicji makra.

Składnia:

   ENDM

Dyrektywa ENDM kończy definicję makra.
Alternatywną nazwą tej dyrektywy jest MACEND.

Predefiniowane symbole.

Rejestry funkcji specjalnych:

Symbol	Nazwa (opis)	Adres
ACC	akumulator	0E0H
B	rejestr B	0F0H
PSW	rejestr stanu	0D0H
SP	wskaźnik stosu	81H
DPTR	dwubajtowy wskaźnik danych	
DPL	młodszy bajt	82H
DPH	starszy bajt	83H
P0	Port 0	80H
P1	Port 1	90H

P2 Port 2 0A0H
P3 Port 3 0B0H
IP rejestr kontroli priorytetów przerwań 0B8H
IE rejestr zezwoleń na przerwania 0A8H
TMOD timery - tryby pracy 89H
TCON timery - sterowanie 88H
TH0 timer 0 starszy bajt 8CH
TL0 timer 0 młodszy bajt 8AH
TH1 timer 1 starszy bajt 8DH
TL1 timer 1 młodszy bajt 8BH
SCON sterownie transmisją szeregową 98H
SBUF bufory transmisji szeregowej 99H
PCON sterowanie trybami 'power down' 87H

Bity adresowalne bezpośrednio

Symbol	Nazwa (opis)	Adres
P	PSW.0	0D0H
OV	PSW.2	0D2H
RS0	PSW.3	0D3H
RS1	PSW.4	0D4H
F0	PSW.5	0D5H
AC	PSW.6	0D6H
CY	PSW.7	0D7H
RXD	P3.0	0B0H
TXD	P3.1	0B1H
INT0	P3.2	0B2H
INT1	P3.3	0B3H
T0	P3.4	0B4H
T1	P3.5	0B5H
WR	P3.6	0B6H
RD	P3.7	0B7H
PX0	IP.0	0B8H
PT0	IP.1	0B9H
PX1	IP.2	0BAH
PT1	IP.3	0BBH
PS	IP.4	0BCH
EX0	IE.0	0A8H
ET0	IE.1	0A9H
EX1	IE.2	0AAH
ET1	IE.3	0ABH
ES	IE.4	0ACH
EA	IE.7	0AFH
IT0	TCON.0	88H
IE0	TCON.1	89H
IT1	TCON.2	8AH
IE1	TCON.3	8BH
TR0	TCON.4	8CH
TF0	TCON.5	8DH
TR1	TCON.6	8EH
TF1	TCON.7	8FH
RI	SCON.0	98H
TI	SCON.1	99H
RB8	SCON.2	9AH
TB8	SCON.3	9BH
REN	SCON.4	9CH
SM2	SCON.5	9DH

SM1 SCON.6 9EH
SM0 SCON.7 9FH

Urządzenia zewnętrzne systemu DSM-51

Symbol	Nazwa (opis)	Adres
CSIC	sterownik przerwań	00H
CSDA	przetwornik cyfrowo/analogowy	08H
CSAD	przetwornik analogowo/cyfrowy	10H
CSMX	multiplekser analogowy	18H
CSKB0	klawiatura matrycowa, klawisze 0..7	21H
CSKB1	klawiatura matrycowa, klawisze 8..	22H
CS55A	układ 8255 rejestr portu A	28H
CS55B	układ 8255 rejestr portu B	29H
CS55C	układ 8255 rejestr portu C	2AH
CS55D	układ 8255 rejestr sterujący	2BH
CSDS	wyświetlacz 7-segm, wybór wskaźnika	30H
CSDB	wyświetlacz 7-segm, bufor danych	38H
CSMOD	dekoder adresów (przełączanie trybu)	40H
LCDWC	wyświetlacz LCD, wpis rozkazów	80H
LCDWD	wyświetlacz LCD, wpis danych	81H
LCDRC	wyświetlacz LCD, odczyt stanu	82H
LCDRD	wyświetlacz LCD, odczyt danych	83H
CSX	zewnętrzna magistrala systemowa	0C0H

Podprogramy standardowe w pamięci EPROM systemu DSM-51

Symbol	Nazwa (opis)	Adres
WRITE_TEXT	wypisanie tekstu na LCD	8100H
WRITE_DATA	wypisanie znaku na LCD	8102H
WRITE_HEX	wypisanie liczby hex na LCD	8104H
WRITE_INSTR	wysłanie rozkazu do LCD	8106H
LCD_INIT	inicjalizacja LCD	8108H
LCD_OFF	wygaszenie LCD	810AH
LCD_CLR	ustawienie w stan początkowy	810CH
DELAY_US	opóźnienie (2*A+6)*12/11.059 us	810EH
DELAY_MS	opóźnienie A ms	8110H
DELAY_100MS	opóźnienie A * 100ms	8112H
WAIT_ENTER	"PRESS ENTER." i czeka na ENTER	8114H
WAIT_ENTER_NW	czekanie na klawisz ENTER	8116H
TEST_ENTER	sprawdzenie klawisza ENTER	8118H
WAIT_ENT_ESC	czekanie na ENTER lub ESC	811AH
WAIT_KEY	czekanie na dowolny klawisz	811CH
GET_NUM	wczytanie liczby BCD (4 cyfry)	811EH
BCD_HEX	zamiana BCD na HEX	8120H
HEX_BCD	zamiana HEX na BCD	8122H
MUL_2_2	mnożenie liczb 2 bajtowych	8124H
MUL_3_1	mnożenie 3bajty * 1bajt	8126H
DIV_2_1	dzielenie 2bajty / 1bajt	8128H
DIV_4_2	dzielenie 4bajty / 2bajty	812AH

Kolejne wersje programu:

Data, wersja	Opis
1994 v 0.90	Asembler dwuprzebiegowy.
12.09.95 v 1.00	Asembler jednoprzebiegowy.

2.10.95 Zmiana opisu: 'Plik wyjściowy:'
v 1.01 na 'Plik wynikowy:'