

Instrukcja użytkownika

Emu8085 – emulator asemblera mikroprocesora 8085

Wszystkie funkcje przycisków z głównej belki programu można również wywoływać z menu lub za pomocą określonych skrótów klawiszowych podanych wewnątrz menu, które zawiera kilka dodatkowych poleceń. Zostały one przedstawione w niniejszym podrozdziale.

1. Menu Plik



Nowy

Tworzy nowy dokument z pustym kodem o nazwie Nowy.txt.



Otwórz...

Otwiera wcześniej zapisany plik z kodem (pliki *.txt).



Zapisz

Zapisuje aktualny kod z okna tekstowego do pliku jako zwykły tekst. Jeżeli nie podano wcześniej nazwy dla nowego pliku zostanie otwarte okno dialogowe z możliwością jej określenia.

Zapisz jako...

Jak wyżej, ale dodatkowo program prosi o podanie nowej nazwy dla zapisywanego pliku.

Zakończ

Kończy działanie aplikacji. Jeżeli ostatnie zmiany w kodzie nie zostały zapisane program zapyta czy je zapisać.

2. Menu Edycja



Wytnij

Usuwa zaznaczony tekst i umieszcza go w schowku (*Clipboard*).



Kopiuj

Kopiuje zaznaczony tekst do schowka.



Wklej

Kopiuje tekst ze schowka do kodu w miejscu kursora tekstowego.



Wyczyść

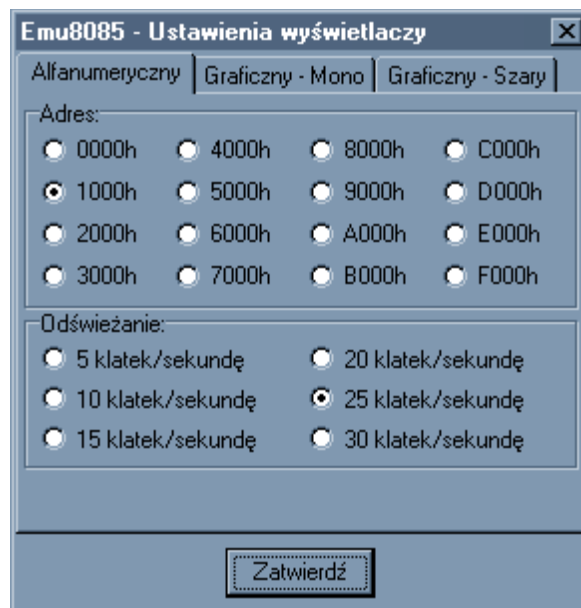
Usuwa zaznaczony tekst.

3. Menu Wyświetlacz

Wyświetlacze posłużyć mogą do prezentowania działania kodu w formie graficznej, co wzbogaca wachlarz możliwości przeprowadzanych symulacji. Obok wyświetlanych na bieżąco rejestrów, pamięci oraz portów, wyświetlacze pełnią rolę uzupełniającą, np. w celu wyświetlania komunikatów bądź formułowania wyników. Program Emu8085 został wyposażony w trzy różne wyświetlacze – alfanumeryczny i dwa graficzne.

Ustawienia...

Otwiera okno konfiguracyjne wyświetlaczy:



Pole **Adres** służy do wyboru komórki pamięci, od której dany wyświetlacz rozpoczyna prezentację jej zawartości. Rodzaj prezentacji zależy od użytego wyświetlacza. Możliwy wybór jednego z 16 podanych adresów.

Pole **Odświeżanie** służy do określenia szybkości odświeżania zawartości wyświetlacza.

Możliwy wybór: od 5 do 30 klatek na sekundę.

Zaznaczenie pola **Wyświetlaj obramowanie pikseli** powoduje dodatkowe obramowanie każdego punktu – piksela czarną ramką (dotyczy wyświetlaczy graficznych).



Alfanumeryczny Włącza lub wyłącza wyświetlacz alfanumeryczny.

Rozmiar: 4 linie po 32 znaki – łącznie 128 znaków wg standardu ASCII. Zawartość wyświetlacza prezentuje 128-bajtowy fragment pamięci począwszy od komórki o adresie zdefiniowanym w oknie **Ustawienia wyświetlaczy** (standardowo 1000h).



Graficzny – Mono Włącza lub wyłącza wyświetlacz graficzny – monochromatyczny.

Rozmiar: 8x8 punktów w kolorze 1-bitowym, gdzie pojedynczy punkt może przyjąć jeden z dwóch stanów: 0 – punkt zgaszony (czarny); 1 – punkt zapalony(biały). Kolejne wiersze pikseli przedstawiają osiem kolejnych bajtów pamięci począwszy od adresu zdefiniowanego w oknie **Ustawienia wyświetlaczy** (standardowo 2000h). Skrajny lewy punkt, piksel reprezentuje stan 0 bitu danej komórki (najmłodszego), drugi piksel – bitu 1, trzeci piksel – bitu 2, itd.



Graficzny – Szary Włącza lub wyłącza wyświetlacz graficzny – w skali szarości.

Rozmiar: 16x16 punktów w 8-bitowej skali szarości: 00h – kolor czarny; FFh – kolor biały. Kolejne piksele przedstawiają 256 komórek pamięci począwszy od adresu zdefiniowanego w oknie **Ustawienia wyświetlaczy** (standardowo 3000h).

4. Menu Symulacja

Zwolnij Zmniejsza o jedną pozycję prędkość przeprowadzanej symulacji.

Przyspiesz Zwiększa o jedną pozycję prędkość przeprowadzanej symulacji.

Prędkość można regulować dodatkowo za pomocą suwaka **Częstotliwość** umieszczonego w prawym górnym rogu głównego okna Emu8085.



Start Uruchamia symulację w trybie ciągłym. Jeśli kod zawiera błędy lub nie spełnia określonych kryteriów symulacja nie zostanie uruchomiona a w oknie **Lista błędów** wyświetli się lista wszystkich błędów wraz z numerem linii, gdzie wystąpiły. W celu uruchomienia symulacji od dowolnego miejsca w kodzie, należy przed jej uruchomieniem wprowadzić do rejestru PC numer linii, od której chcemy rozpocząć symulację.



Krok Uruchamia symulację w trybie krokowym lub, jeśli jest już wstrzymana wykonuje pojedynczą linię kodu i oczekuje na kolejne polecenia użytkownika.



Pauza Wstrzymuje lub kontynuuje symulację w trybie ciągłym. Wstrzymanie symulacji nie powoduje jej zakończenia a jedynie przejście w tryb krokowy.



Stop Kończy symulację lub, jeśli już była zakończona, czyści zawartość rejestrów, pamięci oraz portów – wypełniając je wartościami zerowymi.

5. Główne okno programu

Główne okno programu Emu8085 składa się z:

- menu rozwijanego
- belki z przyciskami, zawierającej najważniejsze polecenia
- okna tekstowego do wprowadzania kodu
- pól z zawartością rejestrów, pamięci, portów oraz flag mikroprocesora
- suwaka do regulacji prędkości symulacji
- okna z listą błędów
- belki dolnej, pokazującej aktualną pozycję kursora tekstowego oraz, w trakcie symulacji, aktualnie wykonywany rozkaz

Zawartość rejestrów, pamięci oraz portów wejściowych można modyfikować w trakcie symulacji poprzez dwukrotne kliknięcie myszą na dany rejestr lub komórkę, co powoduje otwarcie specjalnego okna do wprowadzania danych.

Wprowadzane liczby mogą być podawane w trzech różnych systemach liczbowych: dziesiętnym, dwójkowym, szesnastkowym oraz w formie kodów ASCII. Okno przedstawia dodatkowo reguły wprowadzania liczb:

Dec	Hex	Bin	ASCII
75	4B	01001011	'K'

Reguły wprowadzania liczb:

- Liczby dziesiętne - bez przyrostka, np. 75
- Liczby binarne - z przyrostkiem b, np. 01001011b
- Liczby szesnastkowe - z przyrostkiem h, np. 4Bh
- Znaki ASCII - w apostrofach, np. 'K'

Błędne wprowadzenie liczby lub liczby spoza dopuszczalnego zakresu, jest nie możliwe. Program automatycznie sprawdza poprawność danych i jeśli jest to możliwe przelicza na pozostałe systemy liczbowe (Dec, Hex, Bin, ASCII, co kolejno oznacza: dziesiętny, heksadecymalny - szesnastkowy, binarny – dwójkowy oraz znakowy – ASCII).

6. Objaśnienia dotyczące oznaczeń w programie

- Rejestr flagowy** - jest to rejestr 8-bitowy zawierający stan 5 flag mikroprocesora. Kolejno są to flagi: **S** (bit 7) – znak, liczba ujemna, **Z** (bit 6) – zero, **AC** (bit 4) przeniesienie połówkowe, **P** (bit 2) – parzystość jedynek, **CY** (bit 0) – przeniesienie, pożyczka.
- A, B, C, D, E, H i L**- 8-bitowe rejestry robocze, w tym główny rejestr **A** zwany akumulatorem.
- M** lub **RAM[HL]** - oznacza 8-bitową zawartość komórki pamięci o adresie zawartym w 16-bitowym rejestrze **HL**.
- BC, DE i HL** - 16-bitowe rejestry będące połączeniem pary 8-bitowych rejestrów **B-C**, **D-E** i **H-L**, gdzie pierwsza litera oznacza starszy bajt, tzn. **B**, **D** i **H**.
- PC** - 16-bitowy licznik rozkazów. *UWAGA! W programie zastosowano pewne uproszczenie polegające na tym, że po każdym rozkazie licznik **PC** zwiększa się tylko o 1, nie jak w rzeczywistości od 1 do 3. W programie *Emu8085* jedna linijka kodu to jeden krok licznika rozkazów **PC**. Drugim uproszczeniem jest to, że fizycznie kod nie znajduje się w pamięci (pole Pamięć **RAM**), dzięki czemu uniknięto nieumyślnego uszkodzenia kodu w trakcie działania symulacji.*
- SP** - 16-bitowy wskaźnik stosu.
- /** - za pomocą tego znaku możemy wprowadzać w kodzie komentarze, które będą pomijane w trakcie symulacji – należy postawić go na samym początku każdej linii, w której zamierzamy wpisać jakiś dodatkowy tekst nie będący żadnym rozkazem czy etykietą.
- :** - tym znakiem kończymy nazwy definiowanych etykiet. Etykiety są środkiem zastępczym w zamian za stosowanie adresów bezpośrednich w przypadku instrukcji skoków, takich jak: **JMP**, **CALL** czy **RST**.

Zastosowanie etykiet prezentuje poniższy przykład:

```
MVI A,10
Petla:
    DCR A
    JNZ Petla
HLT
```

Dzięki stosowaniu etykiet, kod programu staje się bardziej przejrzysty i łatwiejszy do późniejszej modyfikacji. Istnieją pewne reguły dotyczące definiowanych etykiet. Po pierwsze: nazwy etykiet mogą mieć maksymalnie 32 znaki, którymi mogą być tylko litery od A do Z (wielkość nie ma znaczenia), cyfry od 0 do 9 oraz znak podkreślenia ‘_’ a pierwszym znakiem musi być litera. Po drugie - etykiety nie mogą się powtarzać. Po trzecie - muszą kończyć się znakiem dwukropka ‘.’. Wszelkie nieprawidłowości dotyczące etykiet będą sygnalizowane w oknie **Lista błędów**.

Rozkaz RST dodatkowo wymaga zdefiniowania etykiet o ściśle określonych nazwach: RST_n, gdzie n jest cyfrą od 0 do 7 określającą numer wywoływanej przez rozkaz procedury. Dzięki temu można samodzielnie napisać kod tych procedur w dowolnym miejscu programu. (Normalnie rozkaz ten wykonuje skok pod określony adres 8·n).

Składnia rozkazów:

- [r] - oznacza jeden z 8-bitowych rejestrów (A, B, C, D, E, H, L lub M).
- [rr] - oznacza jeden z 16-bitowych rejestrów (BC, DE, HL, SP lub PSW – w zależności od rozkazu), w przypadku rejestrów BC, DE i HL podajemy tylko pierwszą literę.
- [aa] - 16-bitowy adres komórki pamięci z zakresu od 0 do 65535 (FFFFh).
- [v] - 8-bitowa liczba z zakresu od 0 do 255 (FFh).
- [vv] - 16-bitowa liczba z zakresu od 0 do 65535 (FFFFh).
- [n] - 3-bitowy numer procedury RST z zakresu od 0 do 7 (7h).

7. Lista rozkazów mikroprocesora 8085

Rozkaz	Flagi	Objaśnienie	Działanie
ADC A	A	Dodawanie z przeniesieniem	A=A+A+CY
ADC B	A	Dodawanie z przeniesieniem	A=A+B+CY
ADC C	A	Dodawanie z przeniesieniem	A=A+C+CY
ADC D	A	Dodawanie z przeniesieniem	A=A+D+CY
ADC E	A	Dodawanie z przeniesieniem	A=A+E+CY
ADC H	A	Dodawanie z przeniesieniem	A=A+H+CY
ADC L	A	Dodawanie z przeniesieniem	A=A+L+CY
ADC M	A	Dodawanie z przeniesieniem	A=A+RAM[HL]+CY
ACI [v]	A	Dodawanie z przeniesieniem	A=A+[v]+CY
ADD A	A	Dodawanie bez przeniesienia	A=A+A
ADD B	A	Dodawanie bez przeniesienia	A=A+B
ADD C	A	Dodawanie bez przeniesienia	A=A+C
ADD D	A	Dodawanie bez przeniesienia	A=A+D
ADD E	A	Dodawanie bez przeniesienia	A=A+E
ADD H	A	Dodawanie bez przeniesienia	A=A+H
ADD L	A	Dodawanie bez przeniesienia	A=A+L
ADD M	A	Dodawanie bez przeniesienia	A=A+RAM[HL]
ADI [v]	A	Dodawanie bez przeniesienia	A=A+[v]
ANA A	D	Iloczyn logiczny	A=A and A
ANA B	D	Iloczyn logiczny	A=A and B
ANA C	D	Iloczyn logiczny	A=A and C
ANA D	D	Iloczyn logiczny	A=A and D
ANA E	D	Iloczyn logiczny	A=A and E
ANA H	D	Iloczyn logiczny	A=A and H
ANA L	D	Iloczyn logiczny	A=A and L
ANA M	D	Iloczyn logiczny	A=A and RAM[HL]
ANI [v]	D	Iloczyn logiczny	A=A and [v]
CALL [aa]	N	Skok ze śladem	PC->stos, PC=aa
CZ	N	...jeśli zero	
CNZ	N	...jeśli nie zero	
CP	N	...jeśli liczba dodatnia	
CM	N	...jeśli liczba ujemna	
CC	N	...jeśli było przeniesienie	
CNC	N	...jeśli nie było przeniesienia	
CPE	N	...jeśli parzysta liczba jedynek	
CPO	N	...jeśli nieparzysta liczba jedynek	
CMA	N	Negacja akumulatora	A=!A
CMC	C	Negacja flagi CY	CY=!CY
CMP A	A	Porównanie A i A	
CMP B	A	Porównanie A i B	
CMP C	A	Porównanie A i C	
CMP D	A	Porównanie A i D	
CMP E	A	Porównanie A i E	
CMP H	A	Porównanie A i H	
CMP L	A	Porównanie A i L	
CMP M	A	Porównanie A i RAM[HL]	
CPI [v]	A	Porównanie A i [v]	

DAA	A	Korekcja dziesiętna akumulatora	
DAD B	C	Dodawanie 16-bitowe	HL=HL+BC
DAD D	C	Dodawanie 16-bitowe	HL=HL+DE
DAD H	C	Dodawanie 16-bitowe	HL=HL+HL
DAD SP	C	Dodawanie 16-bitowe	HL=HL+SP
DCR A	B	Dekrementacja 8-bitowa	A=A-1
DCR B	B	Dekrementacja 8-bitowa	B=B-1
DCR C	B	Dekrementacja 8-bitowa	C=C-1
DCR D	B	Dekrementacja 8-bitowa	D=D-1
DCR E	B	Dekrementacja 8-bitowa	E=E-1
DCR H	B	Dekrementacja 8-bitowa	H=H-1
DCR L	B	Dekrementacja 8-bitowa	L=L-1
DCR M	B	Dekrementacja 8-bitowa	RAM[HL]=RAM[HL]-1
DCX B	N	Dekrementacja 16-bitowa	BC=BC-1
DCX D	N	Dekrementacja 16-bitowa	DE=DE-1
DCX H	N	Dekrementacja 16-bitowa	HL=HL-1
DCX SP	N	Dekrementacja 16-bitowa	SP=SP-1
DI	N	Zablokowanie przerwania INTR	
EI	N	Odblokowanie przerwania INTR	
HLT	N	Zatrzymanie procesora	
IN [v]	N	Odczyt z portu wejściowego	A=IN[v]
INR A	B	Inkrementacja 8-bitowa	A=A+1
INR B	B	Inkrementacja 8-bitowa	B=B+1
INR C	B	Inkrementacja 8-bitowa	C=C+1
INR D	B	Inkrementacja 8-bitowa	D=D+1
INR E	B	Inkrementacja 8-bitowa	E=E+1
INR H	B	Inkrementacja 8-bitowa	H=H+1
INR L	B	Inkrementacja 8-bitowa	L=L+1
INR M	B	Inkrementacja 8-bitowa	RAM[HL]=RAM[HL]+1
INX B	N	Inkrementacja 16-bitowa	BC=BC+1
INX D	N	Inkrementacja 16-bitowa	DE=DE+1
INX H	N	Inkrementacja 16-bitowa	HL=HL+1
INX SP	N	Inkrementacja 16-bitowa	SP=SP+1
JMP [aa]	N	Skok bez śladu	PC=aa
JZ	N	...jeśli zero	
JNZ	N	...jeśli nie zero	
JP	N	...jeśli liczba dodatnia	
JM	N	...jeśli liczba ujemna	
JC	N	...jeśli było przeniesienie	
JNC	N	...jeśli nie było przeniesienia	
JPE	N	...jeśli parzysta liczba jedynek	
JPO	N	...jeśli nieparzysta liczba jedynek	
LDA [aa]	N	Odczyt z pamięci	A=RAM[aa]
LDAX B	N	Odczyt z pamięci	A=RAM[BC]
LDAX D	N	Odczyt z pamięci	A=RAM[DE]
LHLD [aa]	N	Odczyt z pamięci	HL=RAM[aa, aa+1]

LXI B, [vv]	N	Wpisywanie do rejestrów 16-bitow.	BC=[vv]
LXI D, [vv]	N	Wpisywanie do rejestrów 16-bitow.	DE=[vv]
LXI H, [vv]	N	Wpisywanie do rejestrów 16-bitow.	HL=[vv]
LXI SP, [vv]	N	Wpisywanie do rejestrów 16-bitow.	SP=[vv]
MOV A, B	N	Wymiana danych między rejestrami	A=B
MOV A, C	N	Wymiana danych między rejestrami	A=C
MOV A, D	N	Wymiana danych między rejestrami	A=D
MOV A, E	N	Wymiana danych między rejestrami	A=E
MOV A, H	N	Wymiana danych między rejestrami	A=H
MOV A, L	N	Wymiana danych między rejestrami	A=L
MOV A, M	N	Wymiana danych między rejestrami	A=RAM[HL]
MOV B, A	N	Wymiana danych między rejestrami	B=A
MOV B, C	N	Wymiana danych między rejestrami	B=C
MOV B, D	N	Wymiana danych między rejestrami	B=D
MOV B, E	N	Wymiana danych między rejestrami	B=E
MOV B, H	N	Wymiana danych między rejestrami	B=H
MOV B, L	N	Wymiana danych między rejestrami	B=L
MOV B, M	N	Wymiana danych między rejestrami	B=RAM[HL]
MOV C, A	N	Wymiana danych między rejestrami	C=A
MOV C, B	N	Wymiana danych między rejestrami	C=B
MOV C, D	N	Wymiana danych między rejestrami	C=D
MOV C, E	N	Wymiana danych między rejestrami	C=E
MOV C, H	N	Wymiana danych między rejestrami	C=H
MOV C, L	N	Wymiana danych między rejestrami	C=L
MOV C, M	N	Wymiana danych między rejestrami	C=RAM[HL]
MOV D, A	N	Wymiana danych między rejestrami	D=A
MOV D, B	N	Wymiana danych między rejestrami	D=B
MOV D, C	N	Wymiana danych między rejestrami	D=C
MOV D, E	N	Wymiana danych między rejestrami	D=E
MOV D, H	N	Wymiana danych między rejestrami	D=H
MOV D, L	N	Wymiana danych między rejestrami	D=L
MOV D, M	N	Wymiana danych między rejestrami	D=RAM[HL]
MOV E, A	N	Wymiana danych między rejestrami	E=A
MOV E, B	N	Wymiana danych między rejestrami	E=B
MOV E, C	N	Wymiana danych między rejestrami	E=C
MOV E, D	N	Wymiana danych między rejestrami	E=D
MOV E, H	N	Wymiana danych między rejestrami	E=H
MOV E, L	N	Wymiana danych między rejestrami	E=L
MOV E, M	N	Wymiana danych między rejestrami	E=RAM[HL]
MOV H, A	N	Wymiana danych między rejestrami	H=A
MOV H, B	N	Wymiana danych między rejestrami	H=B
MOV H, C	N	Wymiana danych między rejestrami	H=C
MOV H, D	N	Wymiana danych między rejestrami	H=D
MOV H, E	N	Wymiana danych między rejestrami	H=E
MOV H, L	N	Wymiana danych między rejestrami	H=L
MOV H, M	N	Wymiana danych między rejestrami	H=RAM[HL]

MOV L,A	N	Wymiana danych między rejestrami	L=A
MOV L,B	N	Wymiana danych między rejestrami	L=B
MOV L,C	N	Wymiana danych między rejestrami	L=C
MOV L,D	N	Wymiana danych między rejestrami	L=D
MOV L,E	N	Wymiana danych między rejestrami	L=E
MOV L,H	N	Wymiana danych między rejestrami	L=H
MOV L,M	N	Wymiana danych między rejestrami	L=RAM[HL]
MOV M,A	N	Wymiana danych między rejestrami	RAM[HL]=A
MOV M,B	N	Wymiana danych między rejestrami	RAM[HL]=B
MOV M,C	N	Wymiana danych między rejestrami	RAM[HL]=C
MOV M,D	N	Wymiana danych między rejestrami	RAM[HL]=D
MOV M,E	N	Wymiana danych między rejestrami	RAM[HL]=E
MOV M,H	N	Wymiana danych między rejestrami	RAM[HL]=H
MOV M,L	N	Wymiana danych między rejestrami	RAM[HL]=L
MVI A,[v]	N	Wpisywanie do rejestrów 8-bitowych	A=[v]
MVI B,[v]	N	Wpisywanie do rejestrów 8-bitowych	B=[v]
MVI C,[v]	N	Wpisywanie do rejestrów 8-bitowych	C=[v]
MVI D,[v]	N	Wpisywanie do rejestrów 8-bitowych	D=[v]
MVI E,[v]	N	Wpisywanie do rejestrów 8-bitowych	E=[v]
MVI H,[v]	N	Wpisywanie do rejestrów 8-bitowych	H=[v]
MVI L,[v]	N	Wpisywanie do rejestrów 8-bitowych	L=[v]
MVI M,[v]	N	Wpisywanie do rejestrów 8-bitowych	RAM[HL]=[v]
NOP	N	Brak operacji	
ORA A	E	Suma logiczna	A=A or A
ORA B	E	Suma logiczna	A=A or B
ORA C	E	Suma logiczna	A=A or C
ORA D	E	Suma logiczna	A=A or D
ORA E	E	Suma logiczna	A=A or E
ORA H	E	Suma logiczna	A=A or H
ORA L	E	Suma logiczna	A=A or L
ORA M	E	Suma logiczna	A=A or RAM[HL]
ORI [v]	E	Suma logiczna	A=A or [v]
OUT [v]	N	Zapis do portu wyjściowego	OUT[v]=A
PCHL	N	Skok pod adres zawarty w HL	PC=HL
POP B	N	Zdjęcie ze stosu do rejestru	BC<-stos SP=SP+2
POP D	N	Zdjęcie ze stosu do rejestru	DE<-stos SP=SP+2
POP H	N	Zdjęcie ze stosu do rejestru	HL<-stos SP=SP+2
POP PSW	A	Zdjęcie ze stosu A i rej.flagowego	PSW<-stos SP=SP+2
PUSH B	N	Odłożenie rejestru na stos	SP=SP-2 BC->stos
PUSH D	N	Odłożenie rejestru na stos	SP=SP-2 DE->stos
PUSH H	N	Odłożenie rejestru na stos	SP=SP-2 HL->stos
PUSH PSW	N	Odłożenie A i rej.flagow. na stos	SP=SP-2 PSW->stos
RAL	C	Rotacja CY i A w lewo	A=<CY+A<
RAR	C	Rotacja CY i A w prawo	A=>CY+A>
RLC	C	Rotacja A w lewo i do CY	CY=<A A=<A<
RRC	C	Rotacja A w prawo i do CY	CY=>A A=>A>

RIM	N	Odblokowanie przerwania RST	
RET	N	Powrót z podprogramu	PC<-stos
RZ	N	...jeśli zero	
RNZ	N	...jeśli nie zero	
RP	N	...jeśli liczba dodatnia	
RM	N	...jeśli liczba ujemna	
RC	N	...jeśli było przeniesienie	
RNC	N	...jeśli nie było przeniesienia	
RPE	N	...jeśli parzysta liczba jedynek	
RPO	N	...jeśli nieparzysta liczba jedynek	
RST 0	N	Skok do procedury o adresie 00h	PC=00h
RST 1	N	Skok do procedury o adresie 08h	PC=08h
RST 2	N	Skok do procedury o adresie 10h	PC=10h
RST 3	N	Skok do procedury o adresie 18h	PC=18h
RST 4	N	Skok do procedury o adresie 20h	PC=20h
RST 5	N	Skok do procedury o adresie 28h	PC=28h
RST 6	N	Skok do procedury o adresie 30h	PC=30h
RST 7	N	Skok do procedury o adresie 38h	PC=38h
SBB A	A	Odejmowanie z pożyczką	A=A-A-CY
SBB B	A	Odejmowanie z pożyczką	A=A-B-CY
SBB C	A	Odejmowanie z pożyczką	A=A-C-CY
SBB D	A	Odejmowanie z pożyczką	A=A-D-CY
SBB E	A	Odejmowanie z pożyczką	A=A-E-CY
SBB H	A	Odejmowanie z pożyczką	A=A-H-CY
SBB L	A	Odejmowanie z pożyczką	A=A-L-CY
SBB M	A	Odejmowanie z pożyczką	A=A-RAM[HL]-CY
SBI [v]	A	Odejmowanie z pożyczką	A=A-[v]-CY
SHLD [aa]	N	Zapis do pamięci	RAM[aa,aa+1]=HL
SIM	N	Zablokowanie przerwania RST	
SPHL	N	Wpisz do SP zawartość HL	SP=HL
STA [aa]	N	Zapis do pamięci	RAM[aa]=A
STAX B	N	Zapis do pamięci	RAM[BC]=A
STAX D	N	Zapis do pamięci	RAM[DE]=A
STC	C	Ustaw CY na 1	CY=1
SUB A	A	Odejmowanie bez pożyczki	A=A-A
SUB B	A	Odejmowanie bez pożyczki	A=A-B
SUB C	A	Odejmowanie bez pożyczki	A=A-C
SUB D	A	Odejmowanie bez pożyczki	A=A-D
SUB E	A	Odejmowanie bez pożyczki	A=A-E
SUB H	A	Odejmowanie bez pożyczki	A=A-H
SUB L	A	Odejmowanie bez pożyczki	A=A-L
SUB M	A	Odejmowanie bez pożyczki	A=A-RAM[HL]
SUI [v]	A	Odejmowanie bez pożyczki	A=A-[v]
XCHG	N	Zamiana zawartości DE i HL	DE<->HL

XRA A	E	Suma modulo 2	A=A xor A
XRA B	E	Suma modulo 2	A=A xor B
XRA C	E	Suma modulo 2	A=A xor C
XRA D	E	Suma modulo 2	A=A xor D
XRA E	E	Suma modulo 2	A=A xor E
XRA H	E	Suma modulo 2	A=A xor H
XRA L	E	Suma modulo 2	A=A xor L
XRA M	E	Suma modulo 2	A=A xor RAM[HL]
XRI [v]	E	Suma modulo 2	A=A xor [v]
XTHL	N	Zamień wierzchołek stosu z HL	HL<->RAM[SP,SP+1]

Legenda - flagi:

- A - Zmienia wszystkie flagi
- B - Wszystkie oprócz CY
- C - Tylko CY
- D - Wszystkie, CY=0, AC=1
- E - Wszystkie, CY=0, AC=0
- N - Nie zmienia flag

8. Tablica znaków ASCII:

L/H	2	3	4	5	6	7
0		0	@	P	`	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	\$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	,	7	G	W	g	w
8	(8	H	X	h	x
9)	9	I	Y	i	y
A	*	:	J	Z	j	z
B	+	;	K	[k	{
C	,	<	L	\	l	
D	-	=	M]	m	}
E	.	>	N	^	n	~
F	/	?	O	_	o	□