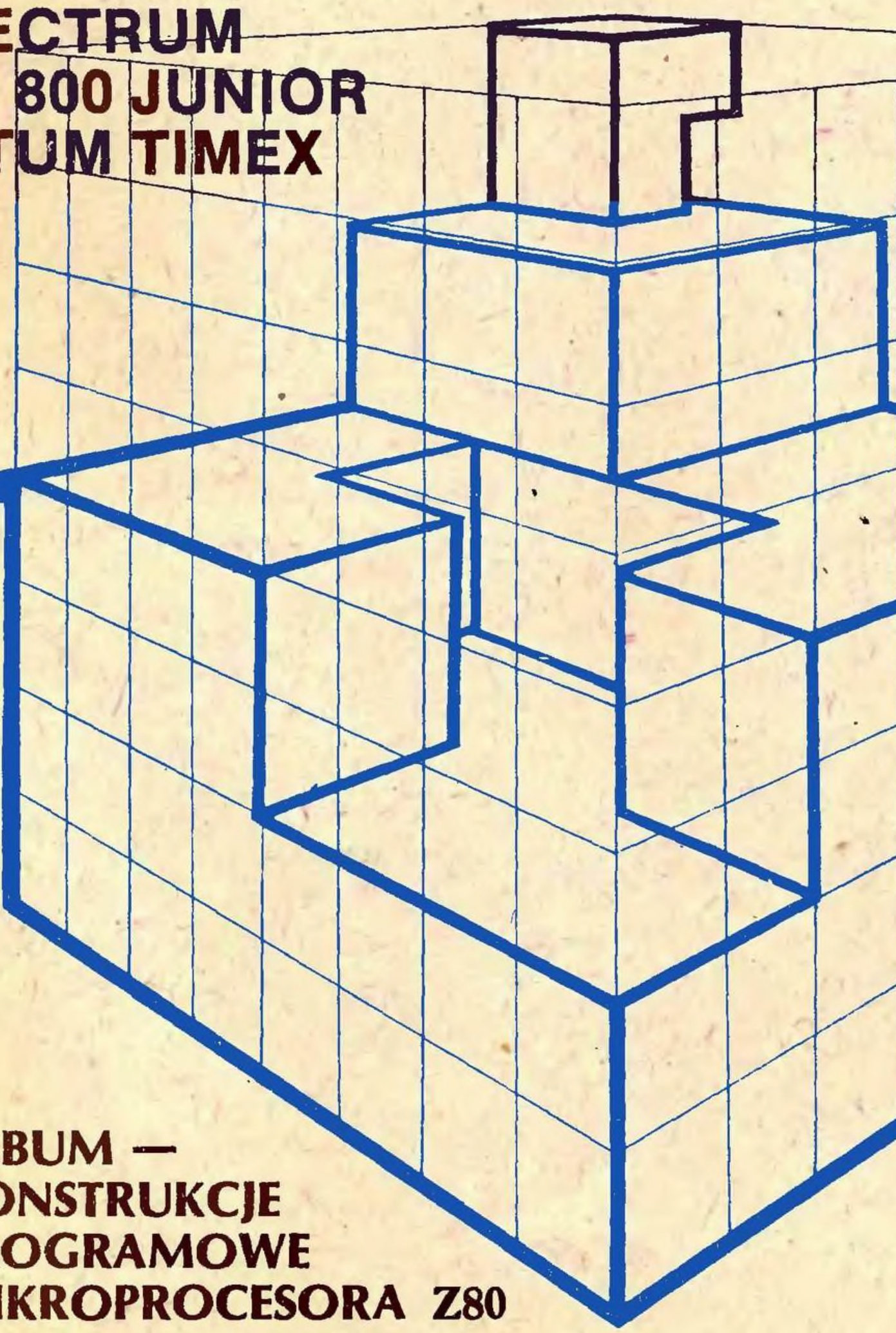
INFORMATYKA
KOMPUTERY
SYSTEMY

Dodatek „Żołnierza Wolności" Cena 150 zł

zeszyty programów komputerowych nr 3

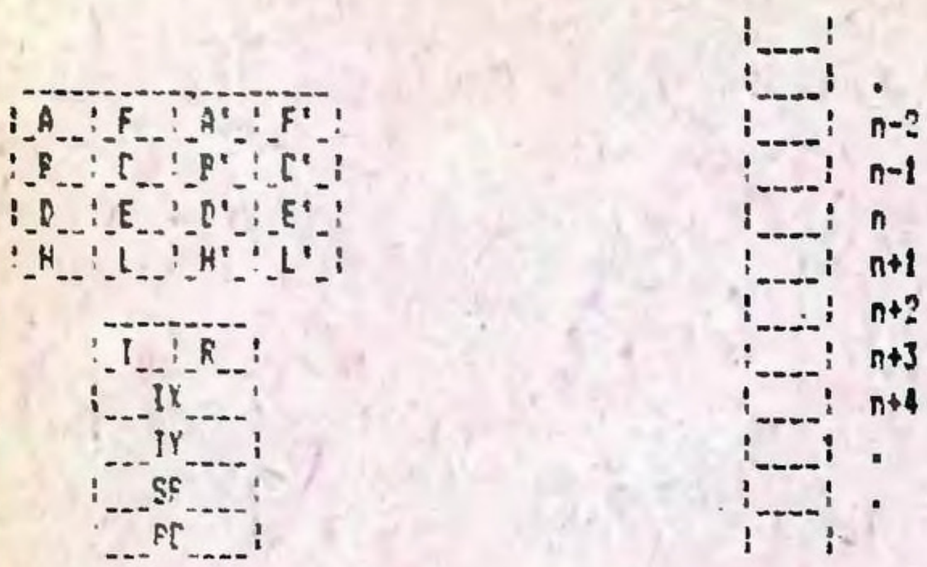
SPECTRUM
ELWRO 800 JUNIOR
MERITUM TIMEX

AMSTRAD
SPECTRA VIDEO
UNIPOLBRIT



ALBUM —
KONSTRUKCJE
PROGRAMOWE
MIKROPROCESORA Z80

PODSTAWOWE KONSTRUKCJE PROGRAMOWE MIKROPROCESORA Z80



(C) P. Ślab, B. Jźwik

SPIS TREŚCI

WPROWADZENIE

Przedstawiamy Czytelnikom album podstawowych konstrukcji programowych, występujących przy kodowaniu programów w języku asemblera dla mikroprocesora Z80. Konstrukcje te można znaleźć niemal w każdym programie, choć nie zawsze są w wyraźny sposób wyodrębnione. Wytrawni programiści z dużą praktyką zawodową znają je "na pamięć". Tajemnica ich dużej wydajności pracy (wyrażanej liczbą uruchomionych i przetestowanych instrukcji w jednostce czasu) wynika z operowania nie tylko pojedyńczymi instrukcjami, ale całymi ciągami instrukcji, realizującymi określone operacje - właśnie konstrukcjami programowymi. Niestety, okres dochodzenia do wprawy trwa stosunkowo długo. Praktycznie każdy programujący jest pozostawiony własnej inwencji. W dostępnej literaturze [2, 3, 4] można wyczytać jedynie opisy realizacji pojedyńczych rozkazów mikroprocesora Z80 i kilku najprostszych konstrukcji programowych. Bardzo trudno jest także "wyciągnąć" tajniki programowania od zawodowych programistów, dla których są to sprawy oczywiste, trywialne i nie warte wspomnienia.

Zadaniem tego materiału jest w jakimś stopniu pomóc początkującym programistom (zarówno zawodowym, jak i amatorom) przez dostarczenie wzorców gotowych konstrukcji programowych. Są one przedstawione w postaci szablonów, które należy przepisać do tekstu programu z ewentualnymi zmianami nazw obszarów danych i etykiet. Oczywiście, nie chodzi tu o zwykłe przepisywanie, konieczne jest zrozumienie zastosowanych rozwiązań. Szczególną uwagę należy zwrócić na warianty tych samych konstrukcji, różniące się głównie czasami realizacji, zajętością pamięci lub rodzajem wykorzystywanych rejestrów.

Przedstawione konstrukcje programowe zostały pogrupowane w następujących 8 rozdziałach:

1. Przesłania pomiędzy rejestrami i komórkami pamięci operacyjnej
2. Pętle iteracyjne
3. Warunki
4. Podprogramy, operacje na stosie
5. Konwersje pól danych
6. Operacje na polach binarnych
7. Operacje arytmetyczne
8. Operacje na tablicach

Dodatki na końcu tekstu zawierają informacje o rozkazach mikroprocesora Z80 i danych liczbowych w różnych systemach liczenia. Jest to materiał, do którego często będziemy zaglądać, szczególnie w początkowym okresie nauki programowania.

Zaproponowany układ treści ma ułatwić odszukanie wymaganych operacji. Przy czym od razu należy się zastrzec, że Czytelnik nie znajdzie wszystkich potrzebnych mu operacji. Liczba możliwych przypadków konstrukcji i ich wariantów jest zbyt duża, aby je znieścić w ograniczonej objętości pisma.

Poszczególne konstrukcje programowe są scharakteryzowane funkcjami chronometrażowymi (t) i pamięciochłonności (p). Funkcja - t, określa czas przebiegu, wyrażony liczbą taktów zegara mikrokomputera, a funkcja p - zajętość pamięci, wyrażoną w bajtach. Wyliczanie czasu przebiegu w mikrosekundach (t') wymaga przekształcenia:

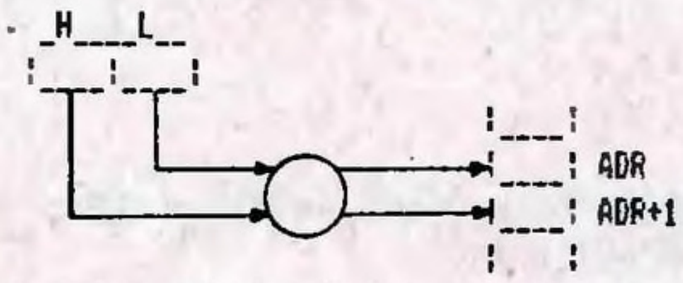
$$t' = t / f \quad [\mu s]$$

gdzie: f - częstotliwość zegara [MHz]

Czasy przebiegu konstrukcji zostały wyznaczone według danych chronometrażowych pojedyńczych rozkazów zestawionych w dodatku I. (kolumna t). W dodatku tym zamieszczono także wyliczone czasy realizacji rozkazów w mikrosekundach dla trzech najczęściej spotykanych częstotliwości zegarów - odpowiednio: 2 MHz, 3.5 MHz i 4 MHz.

Dodatkowo dla celów ilustracyjnych poszczególne konstrukcje programowe zostały opatrzone tzw. "strukturami operacyjnymi", w których występują prostokąty i okręgi połączone liniami skierowanymi. Prostokąty symbolizują obszary danych w pamięci operacyjnej i rejestry, a okręgi - rozkazy. Przy obszarach danych i rejestrach występują ich nazwy. Strzałki przy liniach łączących prostokąty z okręgami wskazują na kierunek przesyłania danych. Odczyt danych jest oznaczony przez zwrot strzałki w kierunku do okręgu, zapis w kierunku do prostokąta symbolizującego obszar danych. Linie ze strzałkami skierowanymi w obydwie strony oznaczają odczyt i zapis.

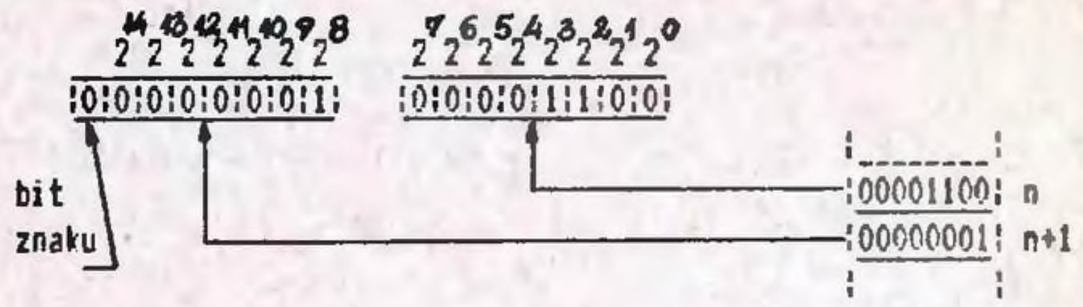
Przykładowo poniższy rysunek ilustruje operację przesyłania zawartości rejestru L do komórki o symbolicznym adresie ADR i rejestru H do komórki o adresie ADR+1. Przyjęto konwencję rysowania komórek pamięci o wzrastających adresach kolejno jedna pod drugą.



Prostokąty przedstawiające rejestry 16-bitowe są dwa razy dłuższe od prostokątów przedstawiających rejestry 8-bitowe. W wypadku pary rejestrów (AF, BC, DE, HL), rejestr zawierający bardziej znaczące pozycje binarne jest narysowany na lewo od rejestru zawierającego mniej znaczące pozycje binarne.

Oprócz linii łączących okręgi z prostokątami (symbolizujące kierunek przepływu danych) na niektórych rysunkach występują także linie skierowane od rejestrów do komórek pamięci operacyjnej. Oznaczają one, że w rejestrach zapisane są adresy komórek (wskaźniki do komórek). Zmiany adresów powstałe w wyniku wykonywania operacji zaznaczone są linią przerywaną.

Wielobajtowe liczby binarne i znakowe w kodzie BCD zapisywane są w kolejnych komórkach pamięci operacyjnej, poczynając od najmniej znaczących pozycji. W ten sposób 2-bajtowa liczba całkowita ze znakiem jest zapamiętana jak niżej:



Wartość dziesiętna liczby zapisanej w komórkach o adresie n i n+1 wynosi 268 (256 + 8 + 4).

Wobec powszechnego stosowania wysokopoziomowych języków algebraicznych (takich jak BASIC, Pascal, C) asembler stosuje się w ściśle określonych przypadkach. Głównie wtedy, gdy czasy realizacji programów kodowanych w języku wysokopoziomowym są zbyt długie. Niewątpliwie najdłuższe czasy występują przy stosowaniu interpretera języka BASIC. W wielu zastosowaniach jest on nie do przyjęcia.

Jak duże różnice uzyskuje się w czasach przebiegu, stosując asembler zamiast BASIC-a, uzmysłowią nam dane chronometrażowe 3 instrukcji podstawienia (na 2-bajtowych zmiennych typu INTEGER) zestawione w poniższej tabeli.

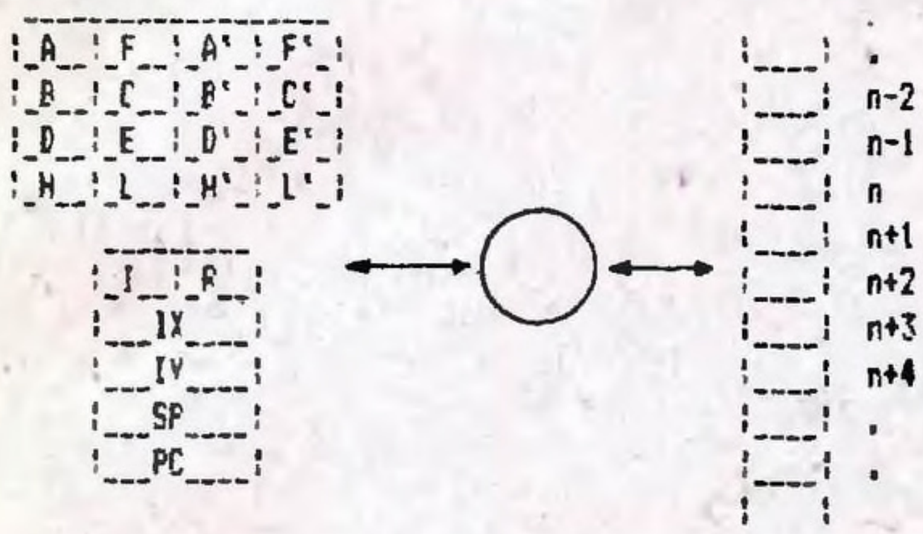
Instrukcja	MERITUM I (1.6 MHz) BASIC [µs]	MERITUM I (1.6 MHz) asembler [µs]	AMSTRAD 6128 (3.3 MHz) BASIC [µs]	AMSTRAD 6128 (3.3 MHz) asembler [µs]
I% = 1	1100	16.3	400	7.9
I% = J%	3100	20	900	10
I% = J%+K%	4500	39.4	1700	19
I% = J%*K%	4800	755	1800	366

Wprawdzie porównanie jest wycinkowe (głębsza analiza wymaga oddzielnego opracowania), ale wyraźnie widać, że stosując oba rodzaje języków, ten sam mikrokomputer staje się narzędziem zupełnie innej klasy. Z mikrokomputerka przekształca się w "poważny" mikrokomputer. Jest to podstawowy powód, dla którego warto opanować niewątpliwie trudną sztukę programowania w języku asemblera.

Portret nowoczesnej szkolnej sekretarki...

1. PRZESŁANIA POMIĘDZY REJESTRAMI I KOMÓRKAMI PAMIĘCI OPERACYJNEJ

Przesłania pomiędzy rejestrami i komórkami pamięci operacyjnej należą do najprostszych, ale najczęściej stosowanych operacji. Zwykle są zapisywane przy użyciu 1, 2 lub 3 rozkazów mikroprocesora. Początkujący programiści miewają sporo kłopotów z ich użyciem ze względu na dużą liczbę rejestrów i specyficzne, asymetryczne własności rozkazów. Termin asymetryczność rozkazów oznacza, że operują tylko na wybranych rejestrach lub parach rejestrów i w różny sposób mogą zmieniać wartość rejestru stanu wskaźników F.



1.1. Przesłania do rejestrów

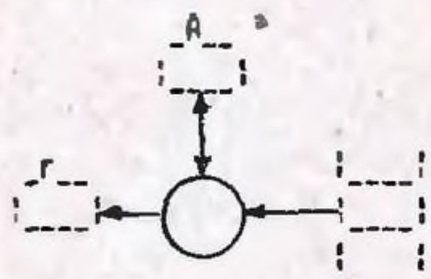
1.1.1. Bezpośrednie przesłania zawartości komórek pamięci operacyjnej do rejestrów

przesłanie zawartości komórki do rejestru A


```
LD   A,(ADR)
```

t = 13, p = 3

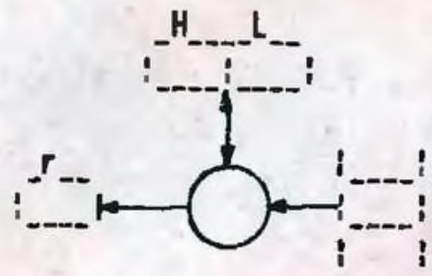
przesłanie zawartości komórki do rejestru roboczego B, C, D, E, H lub L (r) za pośrednictwem rejestru A




```
LD   A,(ADR)
LD   r,A
```

t = 17, p = 4

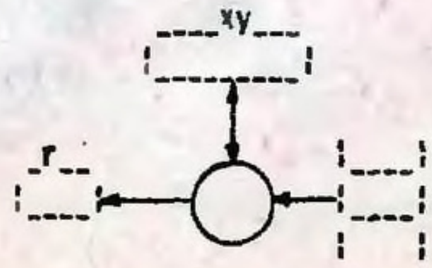
przesłanie zawartości komórki do rejestru roboczego A, B, C, D lub E (r) za pośrednictwem pary rejestrów HL




```
LD   HL,ADR
LD   r,(HL)
```

t = 17, p = 4

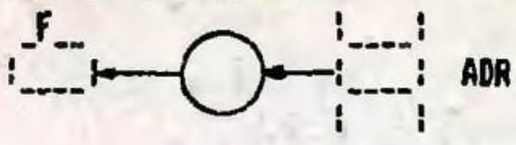
przesłanie zawartości komórki do rejestru roboczego A, B, C, D, E, H lub L (r) za pośrednictwem rejestru IX lub IY (xy)




```
LD   xy,ADR
LD   r,(xy)
```

t = 33, p = 7

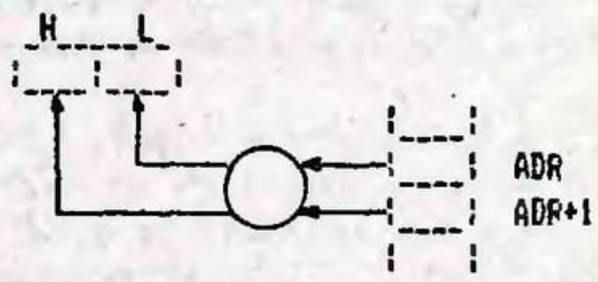
przesłanie zawartości komórki do rejestru znaczników F (zawartość rejestru A zostaje zniszczona)




```
LD   HL,(ADR)
PUSH HL
POP  AF
```

t = 37, p = 5

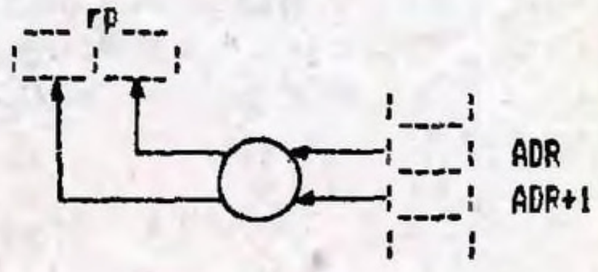
przesłanie zawartości 2 komórek do pary rejestrów HL




```
LD   HL,(ADR)
```

t = 16, p = 3

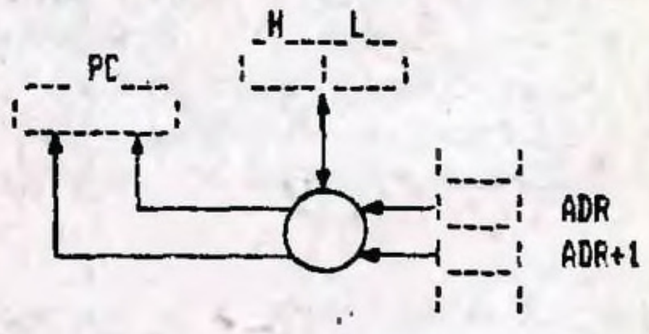
przesłanie zawartości 2 komórek do rejestrów BC, DE, SP, IX lub IY (rp)




```
LD   rp,(ADR)
```

t = 20, p = 4

przesłanie zawartości 2 komórek do licznika rozkazów PC (skok do adresu zapisanego w tych komórkach)

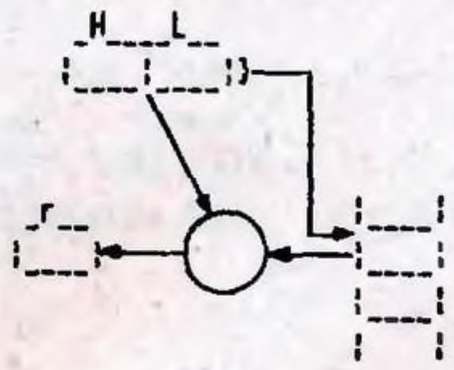



```
LD   HL,(ADR)
JP   (HL)
```

t = 20, p = 4

1.1.2. Pośrednie przesłania zawartości komórek pamięci operacyjnej

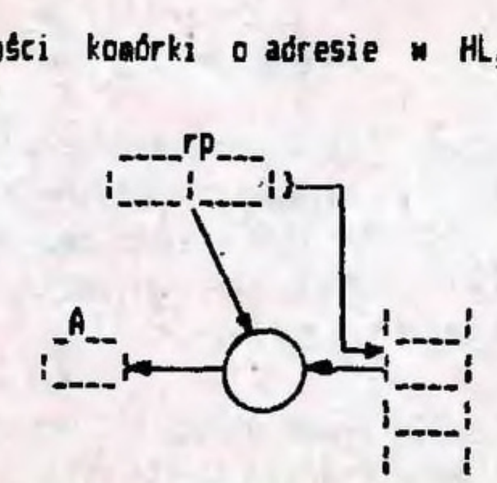
przesłanie zawartości komórki o adresie w HL do rejestru roboczego A, B, C, D lub E (r)




```
LD   r,(HL)
```

t = 7, p = 1

przesłanie zawartości komórki o adresie w HL, BC lub DE (rp) do rejestru A



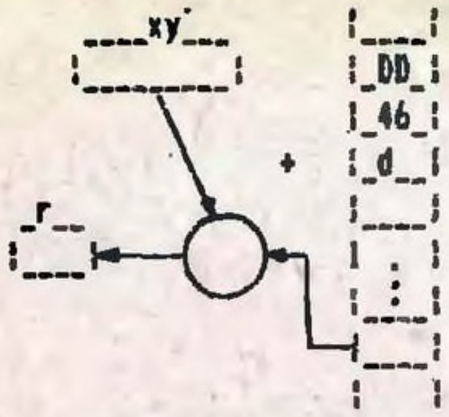

```
LD   A,(rp)
```

t = 7, p = 1

przesłanie zawartości komórki o adresie IX+d lub IY+d (xy+d, d={-127,...,128}) do rejestru roboczego A, B, C, D, E, H lub L (r)

LD r,(xy+d)

t = 19, p = 3

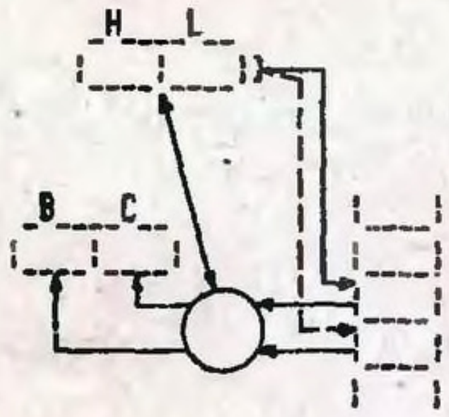


przesłanie zawartości 2 komórek o adresie w HL do pary rejestrów BC lub DE

1.1.3. Przesłania z wierzchołka stosu

LD C,(HL)
INC HL
LD B,(HL)

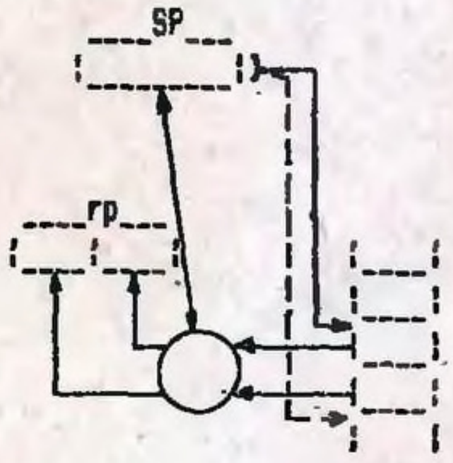
t = 20, p = 3



przesłanie zawartości 2 komórek z wierzchołka stosu do pary rejestrów AF, BC, DE lub HL (rp)

POP rp

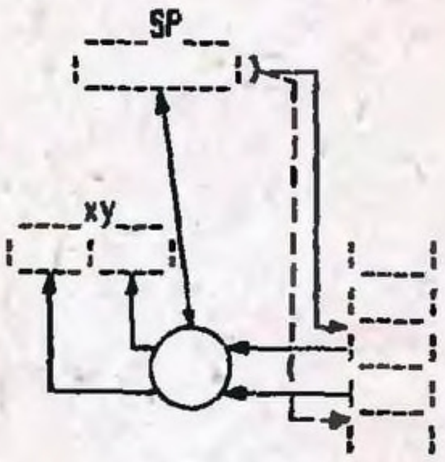
t = 10, p = 1



przesłanie zawartości 2 komórek z wierzchołka stosu do rejestru indeksowego IX lub IY (xy)

POP xy

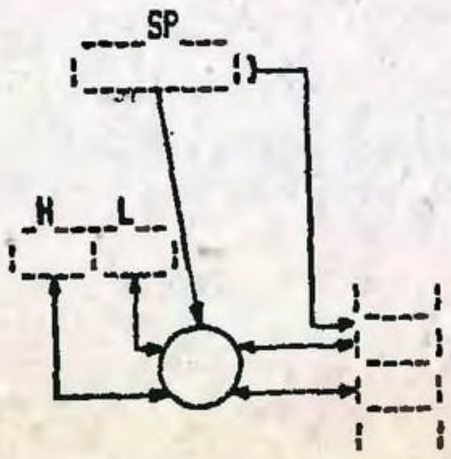
t = 14, p = 2



zamiana miejscami zawartości pary rejestrów HL z zawartością 2 komórek z wierzchołka stosu

EX (SP),HL

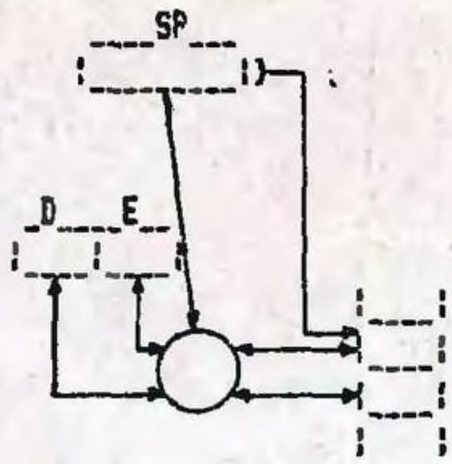
t = 19, p = 1



zamiana miejscami zawartości pary rejestrów DE z zawartością 2 komórek z wierzchołka stosu (bez naruszenia zawartości rejestrów HL)

EX DE,HL
EX (SP),HL
EX DE,HL

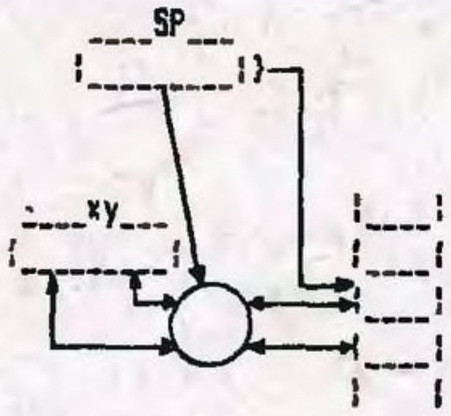
t = 27, p = 3



zamiana miejscami zawartości rejestru indeksowego IX lub IY (xy) z zawartością 2 komórek z wierzchołka stosu

EX (SP),xy

t = 23, p = 2



1.1.4. Przesłania stałych

przesłanie stałej (1 bajt) do rejestru roboczego A, B, C, D, E, H lub L (r)

LD r,86H

t = 7, p = 2

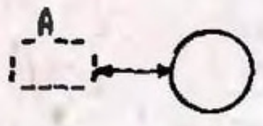
przesłanie 0 do rejestru A

LD A,0

t = 7, p = 2

SUB A

t = 4, p = 1



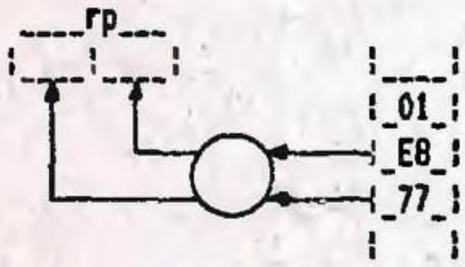
XOR A

t = 4, p = 1

przesłanie stałej (2 bajty) do pary rejestrów BC, DE, HL lub SP (rp)

LD rp,77E8H

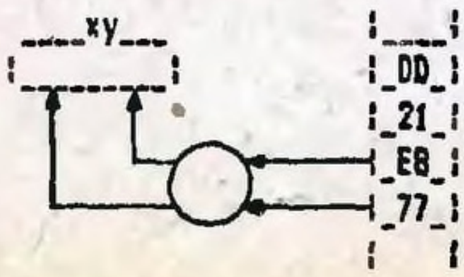
t = 10, p = 3



przesłanie stałej (2 bajty) do rejestru indeksowego IX lub IY (xy)

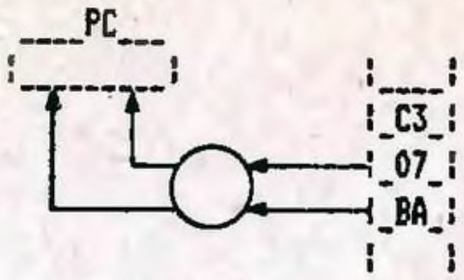
LD xy,77E8H

t = 14, p = 4



przesłanie stałej (2 bajty) do licznika rozkazów PC (skok wg podanego adresu)

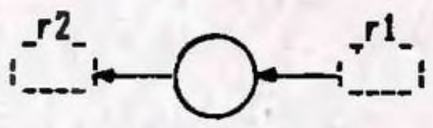
```
JP   BA07H

t = 10, p = 3
```



1.1.5. Przesłania pomiędzy rejestrami

przesłania pomiędzy rejestrami roboczymi A, B, C, D, E, H lub L (z r1 do r2)

```
LD   r2,r1
```



przesłanie zawartości rejestru A do rejestru znaczników F

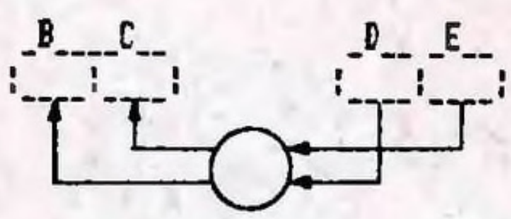

```
LD   H,A
LD   L,A
PUSH HL
POP  AF     ;zawartość rejestru A nie zostaje zniszczona

t = 29, p = 4

PUSH AF
INC  SP
POP  AF
DEC  SP     ;zawartość rejestru A zostaje zniszczona

t = 33, p = 4
```

przesłanie zawartości rejestru znaczników F do rejestru A


```
PUSH AF
POP  BC
LD   A,C    ;zawartość rejestru F nie zostaje zniszczona

t = 25, p = 3

PUSH AF
DEC  SP
POP  AF
INC  SP     ;zawartość rejestru F zostaje zniszczona

t = 33, p = 4
```

przesłania pomiędzy parami rejestrów BC, DE, HL (przykładowo pomiędzy rejestrami BC i DE)

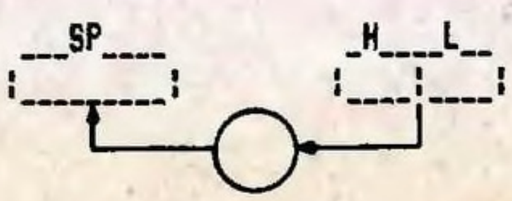
```
LD   C,E
LD   B,D

t = 8, p = 2
```




```
PUSH DE
POP  BC

t = 28, p = 2
```

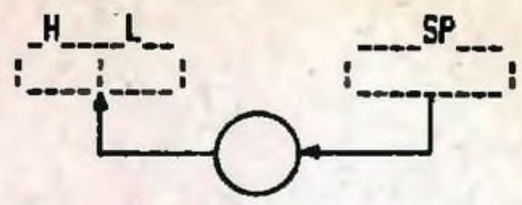
przesłanie zawartości pary rejestrów HL do wskaznika stosu SP

```
LD   SP,HL

t = 6, p = 1
```



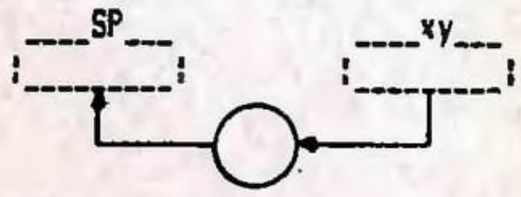
przesłanie zawartości wskaźnika stosu SP do pary rejestrów HL

```
LD   HL,0
ADD  HL,SP

t = 21, p = 4
```



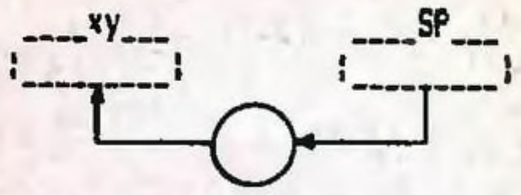
przesłanie zawartości rejestru indeksowego IX lub IY (xy) do wskaźnika stosu SP

```
LD   SP,xy

t = 10, p = 2
```



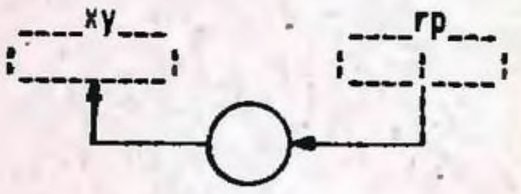
przesłanie zawartości wskaźnika stosu SP do rejestru indeksowego IX lub IY (xy)

```
LD   xy,0
ADD  xy,SP

t = 29, p = 6
```



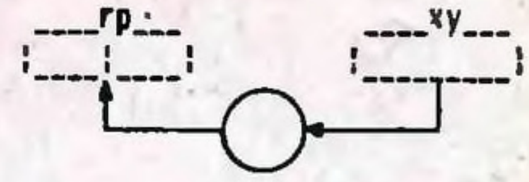
przesłanie zawartości pary rejestrów BC, DE lub HL (rp) do rejestru indeksowego IX lub IY (xy)

```
PUSH rp
POP  xy

t = 25, p = 3
```



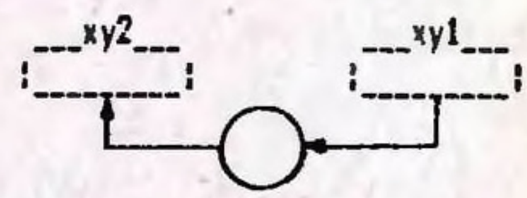
przesłanie zawartości rejestru indeksowego IX lub IY (xy) do pary rejestrów BC, DE lub HL (rp)

```
PUSH xy
POP  rp

t = 25, p = 3
```



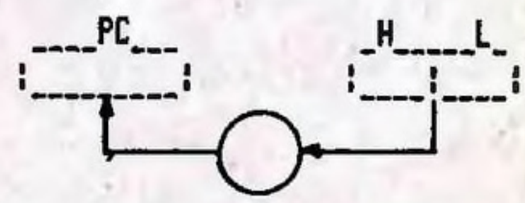
przesłania pomiędzy rejestrami indeksowymi IX, IY (z xy1 do xy2)

```
PUSH xy1
POP  xy2

t = 29, p = 4
```



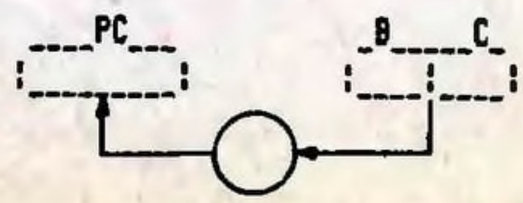
przesłanie zawartości pary rejestrów HL do licznika rozkazów PC (skok do adresu zapisanego w HL)

```
JP   (HL)

t = 4, p = 1
```



przesłanie zawartości pary rejestrów BC do licznika rozkazów PC (skok do adresu zapisanego w BC)

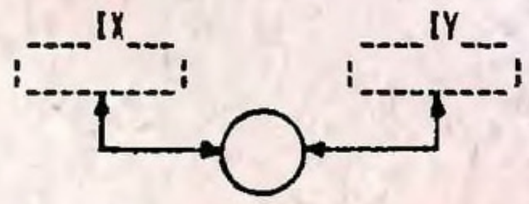
```
PUSH BC
RET

t = 21, p = 2
```



```
PUSH rp
EX   (SP),xy
POP  rp
```

t = 44, p = 4

zamiana miejscami zawartości rejestru indeksowego IX z IY




```
PUSH IX
PUSH IY
POP  IX
POP  IY
```

t = 58, p = 8

```
PUSH IX
EX   (SP),IY
POP  IX
```

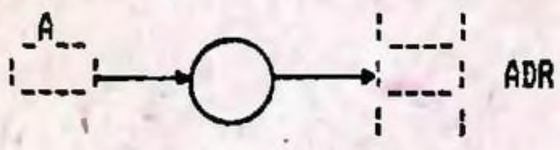
t = 52, p = 6

1.2. Przesłania do komórek pamięci operacyjnej

1.2.1. Bezpośrednie przesłania zawartości rejestrów

przesłanie zawartości akumulatora

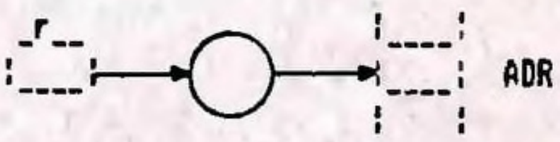
```
LD   (ADR),A
```



t = 13, p = 3

przesłanie zawartości rejestru roboczego B, C, D, E, H lub L (r) za pośrednictwem akumulatora

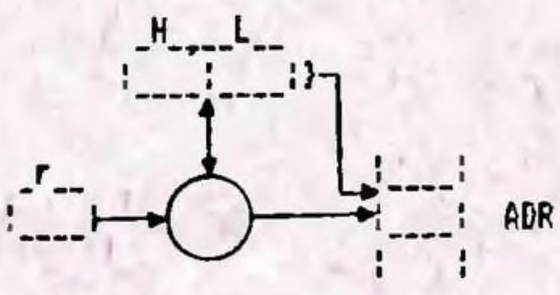
```
LD   A,r
LD   (ADR),A
```



t = 17, p = 4

przesłanie zawartości rejestru roboczego A, B, C, D lub E (r) za pośrednictwem pary rejestrów HL

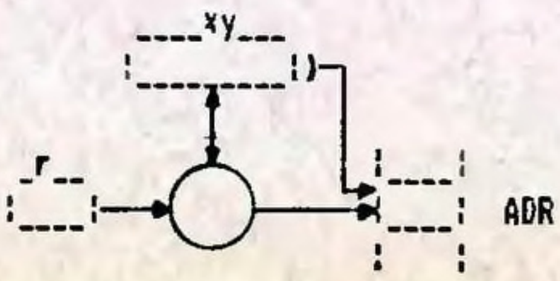
```
LD   HL,ADR
LD   (HL),r
```



t = 17, p = 4

przesłanie zawartości rejestru roboczego A, B, C, D, E, H lub L (r) za pośrednictwem rejestru IX lub IY (xy)

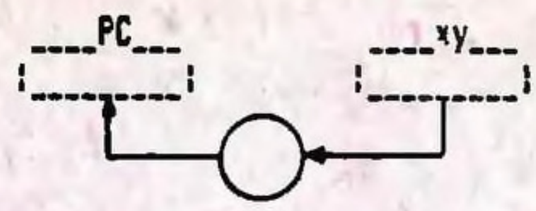
```
LD   xy,ADR
LD   (xy),r
```



t = 33, p = 7

przesłanie zawartości rejestru indeksowego IX lub IY (xy) do licznika rozkazów PC (skok do adresu zapisanego w xy)

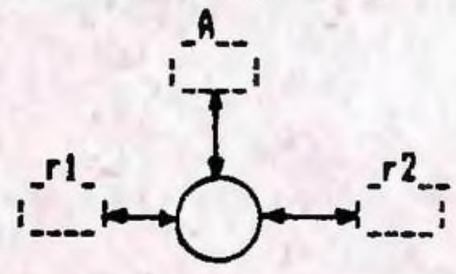
```
JP   (xy)
```



t = 8, p = 2

zamiana miejscami zawartości rejestrów roboczych B, C, D, E, H, L (r1 z r2) za pośrednictwem rejestru A

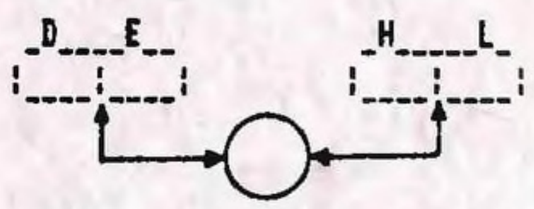
```
LD   A,r1
LD   r1,r2
LD   r2,A
```



t = 17, p = 4

zamiana miejscami zawartości pary rejestrów DE z HL

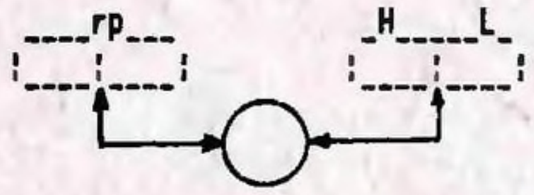
```
EX   DE,HL
```



t = 4, p = 1

zamiana miejscami zawartości pary rejestrów AF, BC lub DE (rp) z parą rejestrów HL

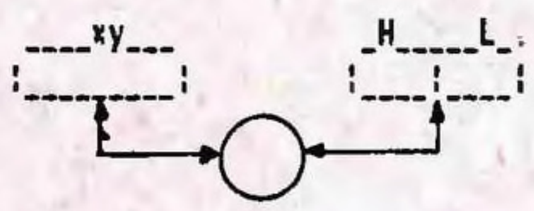
```
PUSH rp
EX   HL,(SP)
POP  rp
```



t = 40, p = 3

zamiana miejscami zawartości rejestrów indeksowych IX lub IY (xy) z parą rejestrów HL

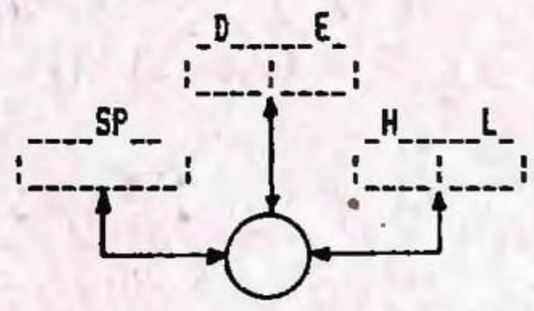
```
PUSH xy
EX   (SP),HL
POP  xy
```



t = 52, p = 6

zamiana miejscami zawartości wskaźnika stosu SP z parą rejestrów HL

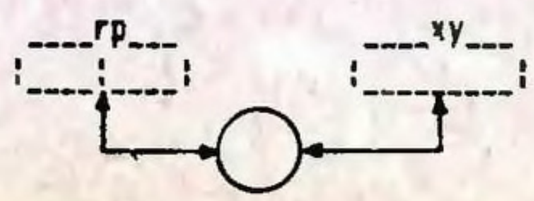
```
EX   DE,HL
LD   HL,0
ADD  HL,SP
EX   DE,HL
LD   SP,HL
EX   DE,HL
```



t = 39, p = 8

zamiana miejscami zawartości pary rejestrów BC, DE lub HL (rp) z rejestrami indeksowymi IX lub IY (xy)

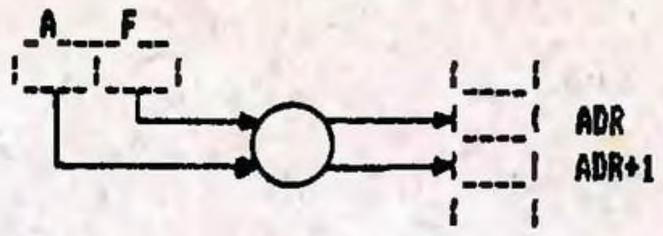
```
PUSH rp
PUSH xy
POP  rp
POP  xy
```



t = 50, p = 6

przesłanie zawartości rejestru znaczników F i akumulatora

PUSH AF
POP HL
LD (ADR),HL

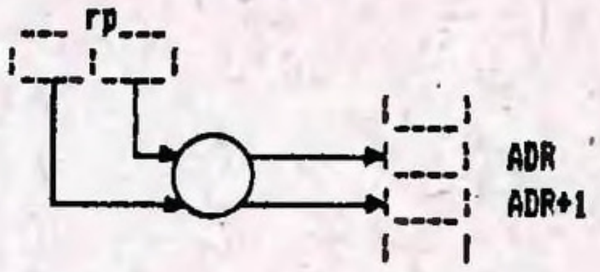


t = 37, p = 5

przesłanie zawartości rejestrów BC, DE, IX, IY lub SP (rp)

LD (ADR),rp

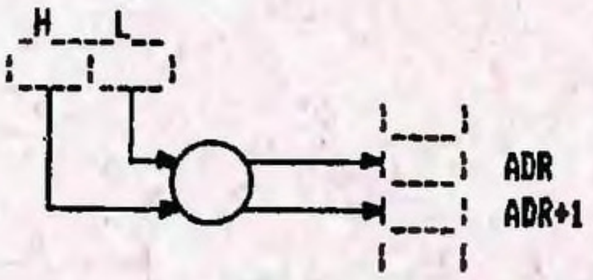
t = 20, p = 4



przesłanie zawartości pary rejestrów HL

LD (ADR),HL

t = 16, p = 3

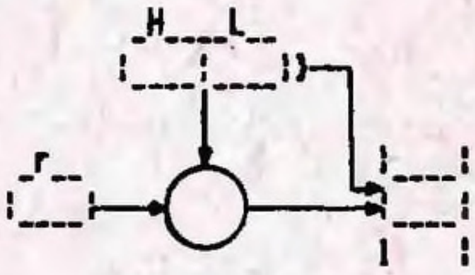


1.2.2. Pośrednie przesłania zawartości rejestrów

przesłanie zawartości rejestru roboczego A, B, C, D, E, H lub L (r) do pamięci o adresie w HL

LD (HL),r

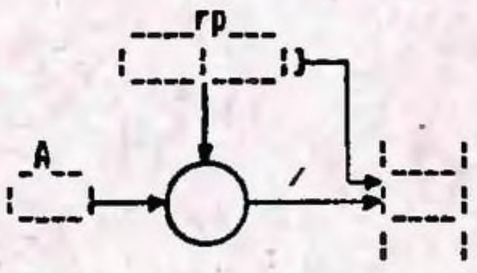
t = 7, p = 1



przesłanie zawartości rejestru A do pamięci o adresie w BC, DE lub HL (rp)

LD (rp),A

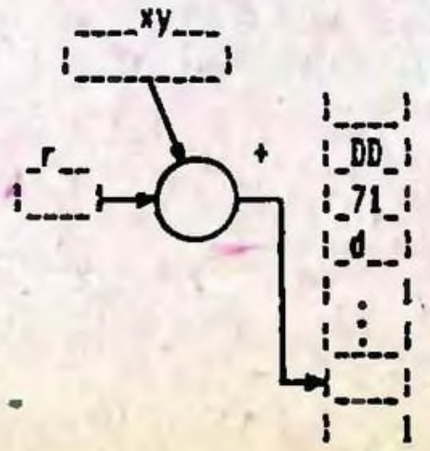
t = 7, p = 1



przesłanie zawartości rejestru A, B, C, D, E, H lub L (r) do pamięci o adresie IX+d lub IY+d (xy+d, d = (-127,...,128))

LD (xy+d),r

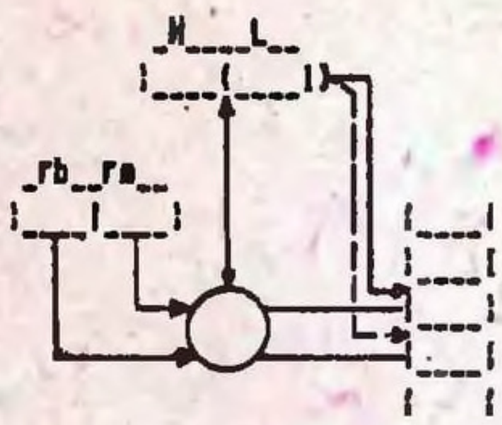
t = 19, p = 3



przesłanie zawartości pary rejestrów BC lub DE (rb = (B, D), rs = (C, E)) do pamięci o adresie w HL

LD (HL),rs
INC HL
LD (HL),rb

t = 20, p = 3

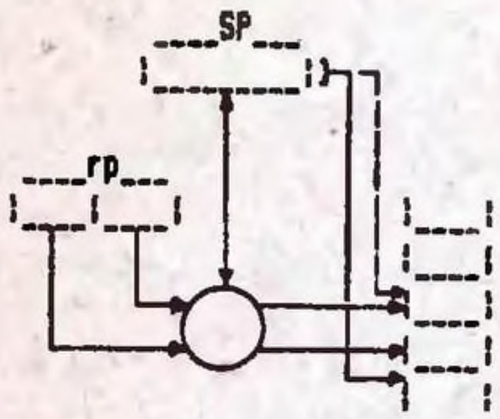


1.2.3. Przesłania na wierzchołek stosu

przesłanie na wierzchołek stosu zawartości pary rejestrów AF, BC, DE lub HL (rp)

PUSH rp

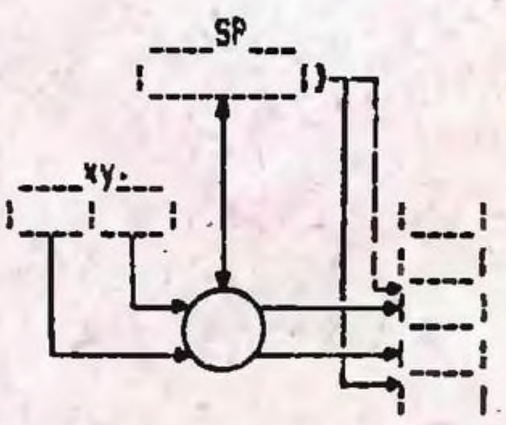
t = 11, p = 1



przesłanie na wierzchołek stosu zawartości rejestru indeksowego IX lub IY (xy)

PUSH xy

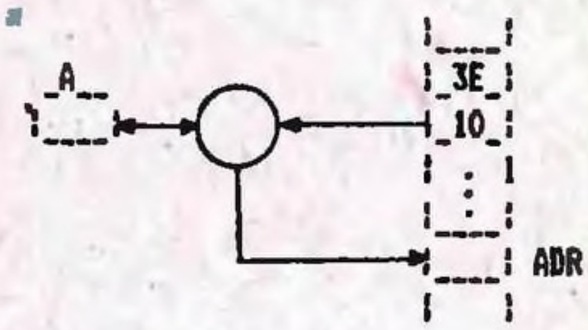
t = 15, p = 2



1.2.4. Przesłania stałych

przesłanie stałej (1 bajt) za pośrednictwem rejestru roboczego A

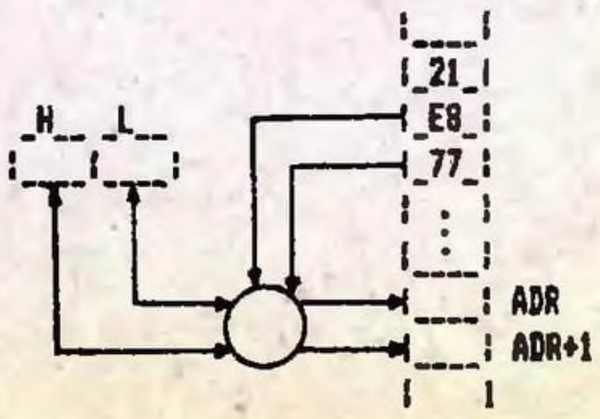
LD A,10H
LD (ADR),A
t = 20, p = 5



przesłanie stałej (2 bajty) za pośrednictwem pary rejestrów HL

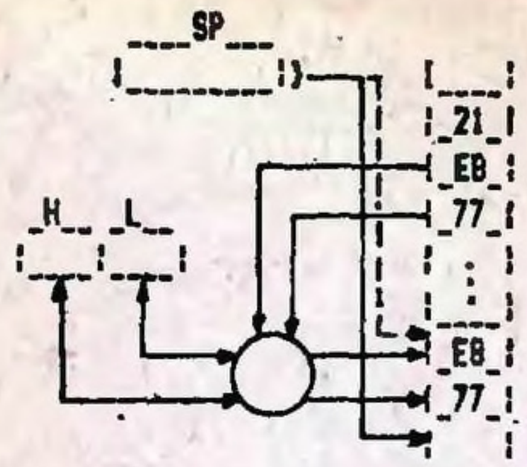
LD HL,77E8H
LD (ADR),HL

t = 26, p = 6



przesłanie stałej (2 bajty) na wierzchołek stosu

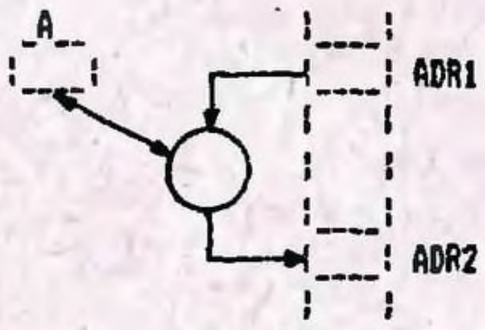
```
LD   HL,77E8H
PUSH HL

t = 21, p = 4

PUSH HL
LD   HL,77E8H
EX   (SP),HL

t = 40, p = 5
```



1.2.5. Przesłania pomiędzy komórkaei paeięci operacyjnej

przesłanie bajtu za pośrednictwem rejestru A

```
LD   A,(ADR1)
LD   (ADR2),A

t = 26, p = 6
```

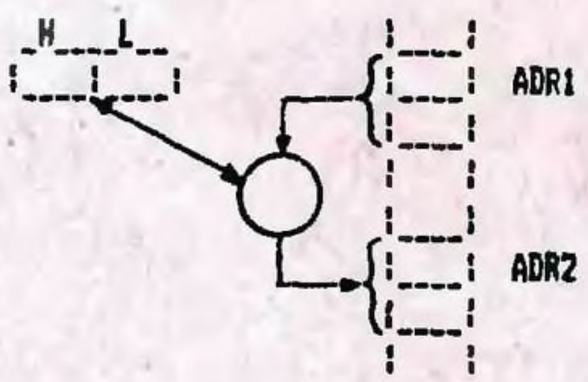


przesłanie bajtu przy użyciu rozkazu LDI lub LDD

```
LD   BC,1                    LD   BC,1
LD   HL,ADR1                 LD   HL,ADR1
LD   DE,ADR2                 LD   DE,ADR2
LDI                          LDD

t = 46, p = 11
```

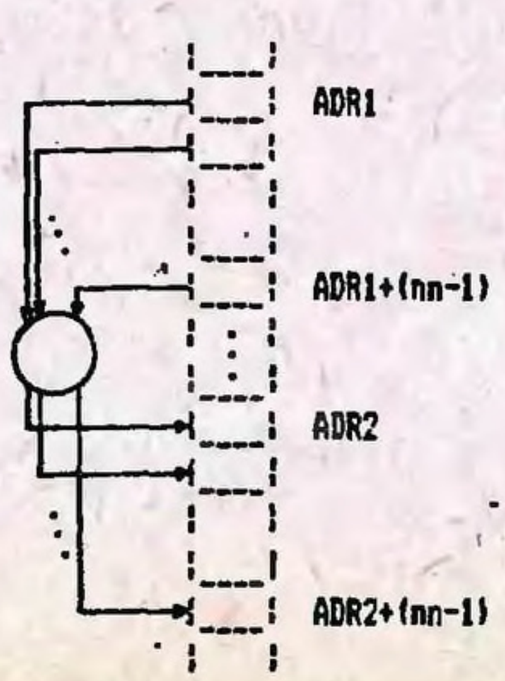
przesłanie 2 bajtów za pośrednictwem pary rejestrów HL oraz BC, DE, SP, IX lub IY (rp)

```
LD   HL,(ADR1)
LD   (ADR2),HL

t = 32, p = 6

LD   rp,(ADR1)
LD   (ADR2),rp

t = 40, p = 8
```



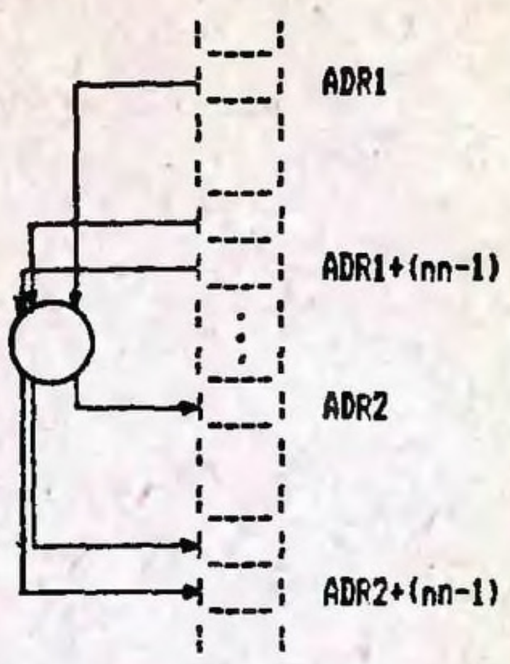
przesłanie bloku nn bajtów przy użyciu rozkazu LDIR [adresy przesyłanych bajtów wzrastają od ADR1 do ADR1+(nn-1)]

```
LD   BC,nn
LD   HL,ADR1
LD   DE,ADR2
LDIR

t = 21*nn + 25,  p = 11
(0 <= nn <= 65355)
```



przesłanie bloku nn bajtów przy użyciu rozkazu LDDR [adresy przesłanych bajtów maleją od ADR1+(nn-1) do ADR1]

```
LD   BC,nn
LD   HL,ADR1+(nn-1)
LD   DE,ADR2+(nn-1)
LDDR

t = 21*nn + 25, p = 11
(0 <= nn <= 65355)
```



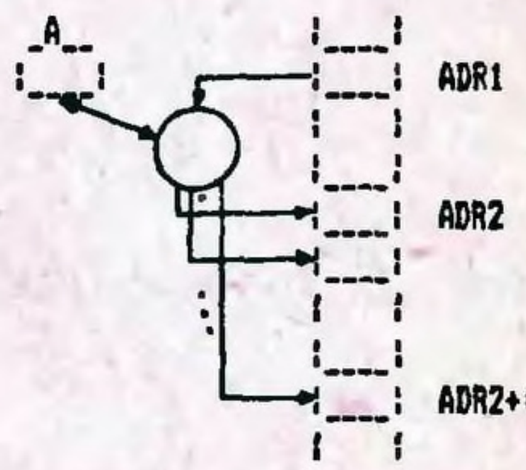
przesłania bloku n i nn bajtów za pośrednictwem rejestru A

```
     LD   HL,ADR1
     LD   DE,ADR2
     LD   B,n
P:   LD   A,(HL)
     LD   (DE),A
     INC  DE
     INC  HL
     DJNZ P

t = 39*n + 22, p = 14
(0 <= n <= 255)
dla n=0 przesłaniu podlega
256 bajtów
```

```
     LD   HL,ADR1
     LD   DE,ADR2
     LD   BC,nn
P:   LD   A,(HL)
     LD   (DE),A
     INC  DE
     INC  HL
     DEC  BC
     LD   A,B
     OR   C
     JR   NI,P

t = nn*52 + 25, p = 18
(0 <= nn <= 65535)
dla nn=0 przesłaniu podlega
65536 bajtów
```

powielenie zawartości komórki

```
     LD   A,(ADR1)
     LD   BC,nn-1
     LD   HL,ADR2
     LD   (HL),A
     LD   DE,ADR2+1
     LDIR

t = 21*nn + 24, p = 15
(0 <= nn <= 65355)
```

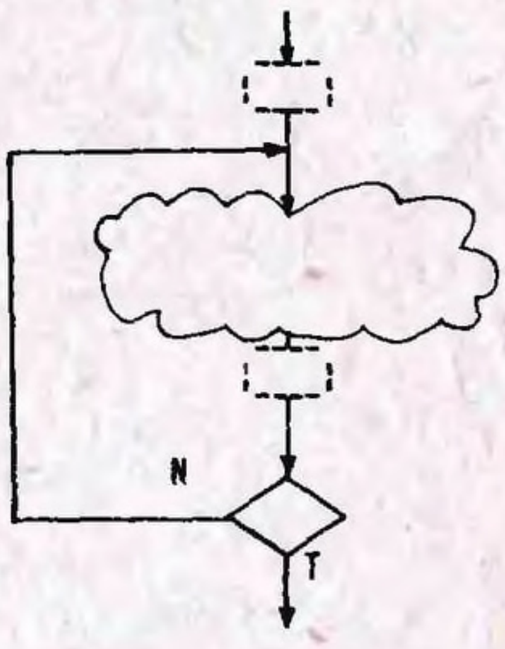



```
     LD   A,(ADR1)
     LD   B,n
     LD   HL,ADR2
P:   LD   (HL),A
     INC  HL
     DJNZ P

t = 26*n + 25, p = 12
(1 <= n <= 255)
dla n=0 powielanie realizowane jest 256 razy
```

2. PĘTLE ITERACYJNE

Najkrótsze czasy przebiegu pętli uzyskuje się dzięki użyciu rozkazu DJNZ i rejestru B, ale tylko dla pętli wykonywanej do 256 razy. Rozkazy mikroprocesora są lepiej przystosowane do kodowania pętli typu "repeat" niż pętli typu "while".

2.1. Pętle typu "repeat"



2.1.1. Pętle wykonywane do 256 razy

Dla n=0 rozkazy objęte pętlą będą wykonywane 256 razy

pętla kodowana przy użyciu rozkazu DJNZ i rejestru B

```
     LD   B,n
P:   .
     .
     .
     DJNZ P
```

t = 13*n + 2, p = 4
(0 <= n <= 255)

pętla kodowana przy użyciu rozkazu DEC i rejestru A, B, C, D, E, H lub L (r)

```
     LD   r,n
P:   .
     .
     .
     DEC  r
     JR   NZ,P
```

t = 16*n + 2, p = 5
(0 <= n <= 255)

pętla kodowana przy użyciu licznika zapamiętanego w komórce pamięci operacyjnej (LICZNIK)

```
     LD   A,n
     LD   (LICZNIK),A
P:   .
     .
     .
     LD   A,(LICZNIK)
     DEC  A
     LD   (LICZNIK),A
     JR   NZ,P
```

t = 42*n + 15, p = 14
(0 <= n <= 255)

pętla gniazdowana

```
     LD   C,n1          ; licznik pętli zewnętrznej
P1:  LD   B,n2          ; licznik pętli wewnętrznej
P2:  .
     .
     .
     DJNZ P2
     DEC  C
     JR   NZ,P1
```

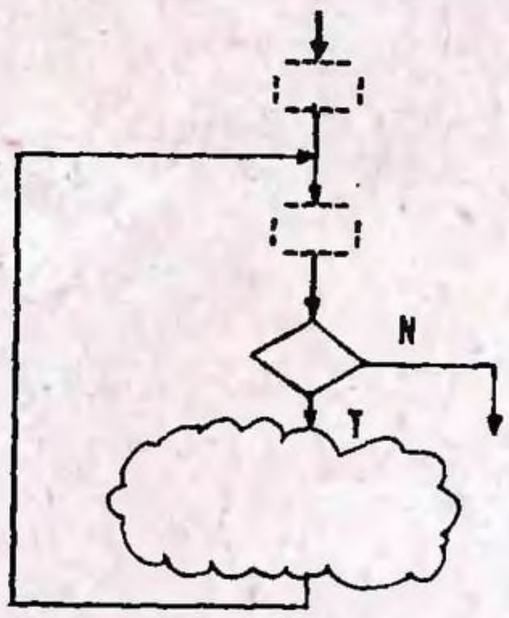
t = 13*n1*n2 + 18*n1 + 2, p = 9
(0 <= n1 <= 255, 0 <= n2 <= 255)
dla n1=0 i n2=0 instrukcje objęte pętlami będą wykonywane 65536 (256*256) razy

2.1.2. Pętla wykonywana do 65536 razy

```
     LD   BC,nn
P:   .
     .
     .
     DEC  BC
     LD   A,B
     OR   C
     JR   NZ,P
```

t = 26*nn + 5 , p = 8
(0 <= nn <= 65535)
dla nn=0 instrukcje objęte pętlą będą wykonywane 65536 razy

2.2. Pętla typy "while"



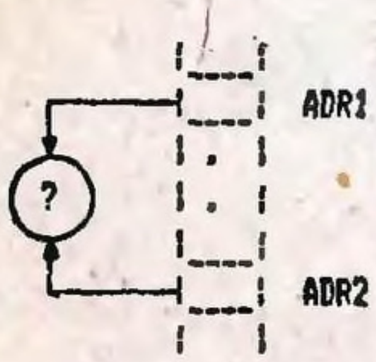
pętla kodowana przy użyciu rozkazu DEC i rejestru A, B, C, D, E, H lub L (r)

```
     LD   r,n+1
P1:  DEC  r
     JR   Z,P2
     .
     .
     .
     JR   P1
P2:  ...
```

t = 23*n + 16, p = 7
(1 <= n <= 254)

3. WARUNKI

3.1. Porównywanie 1-bajtowych tekstów znakowych lub liczb bez znaków



porównywanie zawartości 2 komórek pamięci operacyjnej o adresach ADR1 i ADR2

(ADR1) > (ADR2)
LD A,(ADR2)
LD HL,ADR1
CP (HL)
JR NC,NIE
TAK: ...
t = 37 - tak, 42 - nie, p = 9
	(ADR1) < (ADR2)
	LD A,(ADR1)
	LD HL,ADR2
	CP (HL)
	JR C,TAK
	NIE: ...
	t = 42 - tak, 37 - nie, p = 9

(ADR1) >= (ADR2)
LD A,(ADR1)
LD HL,ADR2
CP (HL)
JR NC,TAK
NIE: ...
t = 42 - tak, 37 - nie, p = 9
	(ADR1) <= (ADR2)
	LD A,(ADR2)
	LD HL,ADR1
	CP (HL)
	JR NC,TAK
	NIE: ...
	t = 42 - tak, 37 - nie, p = 9

(ADR1) = (ADR2)
LD A,(ADR1)
LD HL,ADR2
CP (HL)
JR Z,TAK
NIE: ...
t = 42 - tak, 37 - nie, p = 9
	(ADR1) # (ADR2)
	LD A.(ADR1)
	LD HL,ADR2
	CP (HL)
	JR NZ,TAK
	NIE: ...
	t = 42 - tak, 37 - nie, p = 9

porównywanie zawartości komórki pamięci operacyjnej o adresie ADR ze stałą n

(ADR) > n LD A,(ADR) CP n JR C,NIE JR Z,NIE TAK: ... t = 28 - tak, 26 lub 33 - nie, p = 6	(ADR) < n LD A,(ADR) CP n JR C,TAK NIE: ... t = 26 - tak, 21° - nie, p = 4
(ADR) >= n LD A,(ADR) CP n JR NC,TAK NIE: ... t = 26 - tak, 21 - nie, p = 4	(ADR) <= n LD A,(ADR) CP n JR C,TAK JR Z,TAK NIE: ... t = 26 lub 33 - tak, 28 - nie, p = 6
(ADR) = n LD A,(ADR) CP n JR Z,TAK NIE: ... t = 26 - tak, 21 - nie, p = 4	(ADR) # n LD A,(ADR) CP n JR NZ,TAK NIE: ... t = 26 - tak, 21 - nie, p = 4

n - wielkość 8-bitowa

porównywanie zawartości rejestru A, B, C, D, E, H lub L (r) ze stałą n

r > n LD A,n CP r JR NC,NIE TAK: ... t = 18 - tak, 23 - nie, p = 5	r < n LD A,r CP n JR C,TAK NIE: ... t = 23 - tak, 18 - nie, p = 5
r >= n LD A,r CP n JR NC,TAK NIE: ... t = 23 - tak, 18 - nie, p = 4	r <= n LD A,n CP r JR NC,TAK NIE: ... t = 23 - tak, 18 - nie, p = 6
r = n LD A,r CP n JR Z,TAK NIE: ... t = 23 - tak, 18 - nie, p = 5	r # n LD A,r CP n JR NZ,TAK NIE: ... t = 23 - tak, 18 - nie, p = 5

3.2. Porównywanie 1-bajtowych liczb całkowitych ze znakiem

porównywanie zawartości 2 komórek pamięci operacyjnej o adresach ADR1 i ADR2

(ADR1) > (ADR2)	LD	A,(ADR1)	(ADR1) < (ADR2)	LD	A,(ADR1)
	LD	HL,ADR2		LD	HL,ADR2
	CP	(HL)		CP	(HL)
	JR	Z,NIE		JP	M,P1
	JP	M,P1		JP	PE,TAK
	JP	PO,TAK		JR	NIE
	JR	NIE	P1:	JP	PO,TAK
P1:	JP	PE,TAK	NIE:	...	
NIE:	...				
t = 57 - tak, 42,57 lub 69 - nie, p = 20			t = 50 - tak, 50 lub 62 - nie, p = 18		
(ADR1) >= (ADR2)	LD	A,(ADR1)	(ADR1) <= (ADR2)	LD	A,(ADR1)
	LD	HL,ADR2		LD	HL,ADR2
	CP	(HL)		CP	(HL)
	JP	M,P1		JR	Z,TAK
	JP	PO,TAK		JP	M,P1
	JR	NIE		JP	PE,TAK
P1:	JP	PE,TAK		JR	NIE
NIE:	...		P1:	JP	PO,TAK
			NIE:	...	
t = 50 - tak, 50 lub 62 - nie, p = 18			t = 42 lub 57 - tak, 57 lub 69 - nie, p = 20		
(ADR1) = (ADR2)	LD	A,(ADR1)	(ADR1) # (ADR2)	LD	A,(ADR1)
	LD	HL,ADR2		LD	HL,ADR2
	CP	(HL)		CP	(HL)
	JR	Z,TAK		JR	NZ,TAK
NIE:	...		NIE:	...	
t = 42 - tak, 37 - nie, p = 9			t = 42 - tak, 37 - nie, p = 9		

3.3. Porównywanie 2-bajtowych liczb bez znaku ze stałą nn

porównywanie stałej nn z liczbą zapisaną w komórkach pamięci operacyjnej o adresach ADR i ADR+1

(ADR) > nn	LD	HL,(ADR)	(ADR) < nn	LD	HL,(ADR)
	LD	BC,nn		LD	BC,nn
	OR	A		OR	A
	SBC	HL,BC		SBC	HL,BC
	JR	C,NIE		JR	C,TAK
	JR	Z,NIE		JR	Z,TAK
TAK:	...		NIE:		
t = 63 - tak, 61 lub 68 - nie, p = 14			t = 61 lub 68 - tak, 63 - nie, p = 14		
(ADR) >= nn	LD	HL,(ADR)	(ADR) <= nn	LD	HL,(ADR)
	LD	BC,nn		LD	BC,nn
	OR	A		OR	A
	SBC	HL,BC		SBC	HL,BC
	JR	NC,TAK		JR	C,TAK
NIE:	...			JR	Z,TAK
			NIE:		
t = 61 - tak, 56 - nie, p = 12			t = 61 lub 68 - tak, 63 - nie, p = 14		
(ADR) = nn	LD	HL,(ADR)	(ADR) # nn	LD	HL,(ADR)
	LD	BC,nn		LD	BC,nn
	OR	A		OR	A
	SBC	HL,BC		SBC	HL,BC
	JR	Z,TAK		JR	NZ,TAK
NIE:	...		NIE:	...	
t = 61 - tak, 56 - nie, p = 12			t = 61 - tak, 56 - nie, p = 12		

Rozkazy OR A, użyte w powyższej tablicy i trzech następnych powodują wyzerowanie wskaźnika przeniesienia CY, ten sam efekt można uzyskać przez zastosowanie sekwencji rozkazów SCF i CCF.
nn - 16-bitowa liczba bez znaku

porównywanie stałej nn z zawartością pary rejestrów DE

DE > nn	LD	HL,nn	DE < nn	LD	L,E
	OR	A		LD	H,D
	SBC	HL,DE		LD	BC,nn
	JR	NC,NIE		OR	A
TAK:	...			SBC	HL,BC
				JR	C,TAK
			NIE:	...	
t = 36 - tak, 41 - nie, p = 8			t = 49 - tak, 44 - nie, p = 10		
DE >= nn	LD	L,E	DE <= nn	LD	HL,nn
	LD	H,D		OR	A
	LD	BC,nn		SBC	HL,DE
	OR	A		JR	NC,TAK
	SBC	HL,BC	NIE:	...	
	JR	NC,TAK			
NIE:	...				
t = 49 - tak, 44 - nie, p = 10			t = 41 - tak, 36 - nie, p = 8		
DE = nn	LD	HL,nn	DE # nn	LD	HL,nn
	OR	A		OR	A
	SBC	HL,DE		SBC	HL,DE
	JR	Z,TAK		JR	NZ,TAK
NIE:	...		NIE:	...	
t = 41 - tak, 36 - nie, p = 8			t = 41 - tak, 36 - nie, p = 8		

porównywanie stałej nn z zawartością rejestru SP

```
SP > nn          LD  HL,nn        SP < nn          LD  HL,0
                 OR  A                             ADD HL,SP
                 SBC HL,SP                         LD  BC,nn
                 JR  NC,NIE                        OR  A
           TAK:  ...                               SBC HL,BC
                                                   JR  C,TAK
                                             NIE:  ...

t = 36 - tak , p = 8 / 41 - nie          t = 62 - tak , p = 12 / 57 - nie

SP >= nn         LD  HL,0         SP <= nn         LD  HL,nn
                 ADD HL,SP                         OR  A
                 LD  BC,nn                         SBC HL,SP
                 OR  A                             JR  NC,TAK
                 SBC HL,BC                   NIE:  ...
                 JR  NC,TAK
           NIE:  ...

t = 62 - tak , p = 12 / 57 - nie         t = 41 - tak , p = 8 / 36 - nie

SP = nn          LD  HL,nn        SP # nn          LD  HL,nn
                 OR  A                             OR  A
                 SBC HL,SP                         SBC HL,SP
                 JR  Z,TAK                         JR  NZ,TAK
           NIE:  ...                         NIE:  ...

t = 41 - tak , p = 8 / 36 - nie          t = 41 - tak , p = 8 / 36 - nie
```

nn - 16-bitowa liczba bez znaku

porównywanie stałej nn z zawartością pary rejestrów HL

```
HL > nn          LD  DE,nn        HL < nn          PUSH HL
                 EX  DE,HL                         LD  DE,nn
                 OR  A                             OR  A
                 SBC HL,DE                         SBC HL,DE
                 EX  DE,HL                         POP HL
                 JR  NC,NIE                        JR  C,TAK
           TAK:  ...                         NIE:  ...

t = 44 - tak , p = 10 / 49 - nie         t = 62 - tak , p = 10 / 57 - nie

HL >= nn         PUSH HL          HL <= nn         LD  DE,nn
                 LD  DE,nn                         EX  DE,HL
                 OR  A                             OR  A
                 SBC HL,DE                         SBC HL,DE
                 POP HL                            EX  DE,HL
                 JR  NC,TAK                        JR  NC,TAK
           NIE:  ...                         NIE:

t = 62 - tak , p = 10 / 57 - nie         t = 49 - tak , p = 10 / 44 - nie
```

```
HL = nn          LD  DE,nn        HL # nn          LD  DE,nn
                 EX  DE,HL                         EX  DE,HL
                 OR  A                             OR  A
                 SBC HL,DE                         SBC HL,DE
                 EX  DE,HL                         EX  DE,HL
                 JR  Z,TAK                         JR  NZ,TAK
           NIE:  ...                         NIE:  ...

t = 49 - tak , p = 10 / 44 - nie         t = 49 - tak , p = 10 / 44 - nie
```

nn - 16-bitowa liczba bez znaku

3.4. Porównywanie n-bajtowych tekstów znakowych

porównywanie tekstów znakowych, zapisanych w komórkach począwszy od adresów ADR1 i ADR2

```
ADR1 > ADR2      LD  DE,ADR1      ADR1 < ADR2      LD  DE,ADR1
                 LD  HL,ADR2                       LD  HL,ADR2
                 LD  B,n                           LD  B,n
           P1:   LD  A,(DE)                  P1:   LD  A,(DE)
                 CP  (HL)                          CP  (HL)
                 JR  NZ,P2                         JR  NZ,P2
                 INC DE                            INC DE
                 INC HL                            INC HL
                 DJNZ P1                           DJNZ P1
                 JR  NIE                           JR  NIE
           P2:   JR  C,NIE                   P2:   JR  C,TAK
           TAK:  ...                         NIE:  ...

p = 19                                     p = 19

ADR1 >= ADR2     LD  DE,ADR1      ADR1 <= ADR2     LD  DE,ADR1
                 LD  HL,ADR2                       LD  HL,ADR2
                 LD  B,n                           LD  B,n
           P1:   LD  A,(DE)                  P1:   LD  A,(DE)
                 CP  (HL)                          CP  (HL)
                 JR  NZ,P2                         JR  NZ,P2
                 INC DE                            INC DE
                 INC HL                            INC HL
                 DJNZ P1                           DJNZ P1
           TAK:  ...                         TAK:  ...
           P2:   JR  NC,TAK                  P2:   JR  C,TAK
           NIE:  ...                         NIE:  ...

p = 17                                     p = 17

ADR1 = ADR2      LD  DE,ADR1      ADR1 # ADR2      LD  DE,ADR1
                 LD  HL,ADR2                       LD  HL,ADR2
                 LD  B,n                           LD  B,n
           P1:   LD  A,(DE)                  P1:   LD  A,(DE)
                 CP  (HL)                          CP  (HL)
                 JR  NZ,NIE                        JR  NZ,TAK
                 INC DE                            INC DE
                 INC HL                            INC HL
                 DJNZ P1                           DJNZ P1
           TAK:  ...                         NIE:  ...

p = 15                                     p = 15
```

(0 <= n <= 255)
dla n=0 porównywane są teksty o długości 256 znaków

czasy przebiegu operacji porównywania n-bajtowych tekstów znakowych dla przypadków spełnienia i nie spełnienia warunków

tak	nie
ADR1 > ADR2	
t = 46tk + 14	t = 46tk + 19 gdy ADR1 < ADR2 t = 46tn + 34 gdy ADR1 = ADR2
ADR1 < ADR2	
t = 46tk + 19	t = 46tk + 14 gdy ADR1 > ADR2 t = 46tn + 34 gdy ADR1 = ADR2
ADR1 >= ADR2	
t = 46tk + 19 gdy ADR1 > ADR2 t = 46tn + 22 gdy ADR1 = ADR2	t = 46tk + 19
ADR1 <= ADR2	
t = 46tk + 19 gdy ADR1 < ADR2 t = 46tn + 22 gdy ADR1 = ADR2	t = 46tk + 19
ADR1 = ADR2	
t = 46tn + 22	t = 46tk + 7
ADR1 # ADR2	
t = 46tk + 7	t = 46tn + 22

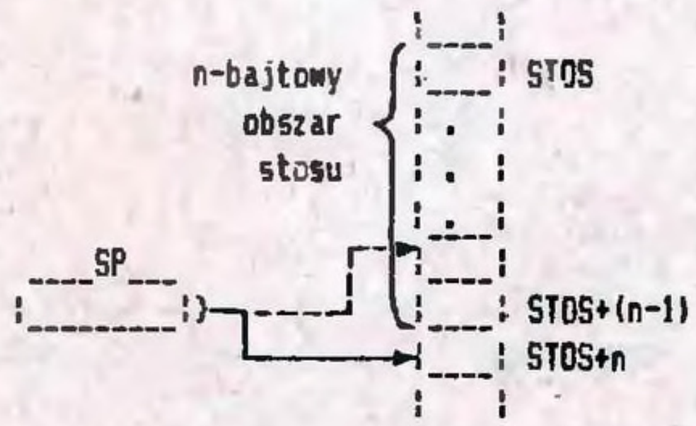
k - liczba porównań znaków (1 <= k <= n)

4. PODPROGRAMY, OPERACJE NA STOSIE

Operacje wykonywane na elementach stosu muszą być poprzedzone ustaleniem wartości początkowej rejestru wskaźnika stosu - SP. Dla n-bajtowego obszaru stosu, rozpoczynającego się od symbolicznego adresu STOS, realizowane jest to rozkazem:

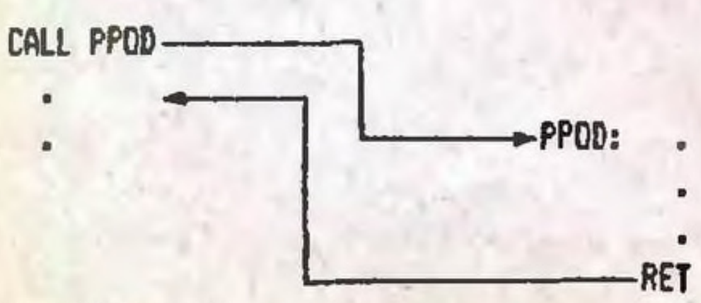
```
LD   SP,STOS+n
```

t = 10, p = 3



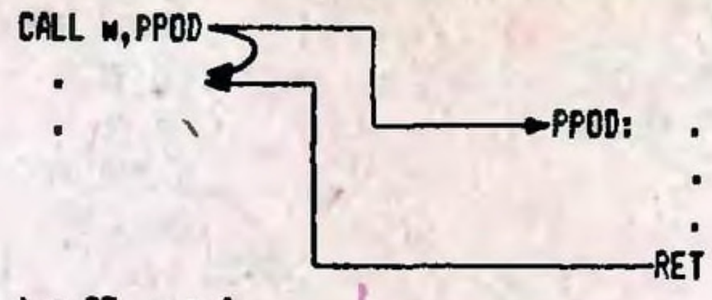
4.1. Wywoływanie podprogramów

wywołanie i powrót z podprogramu



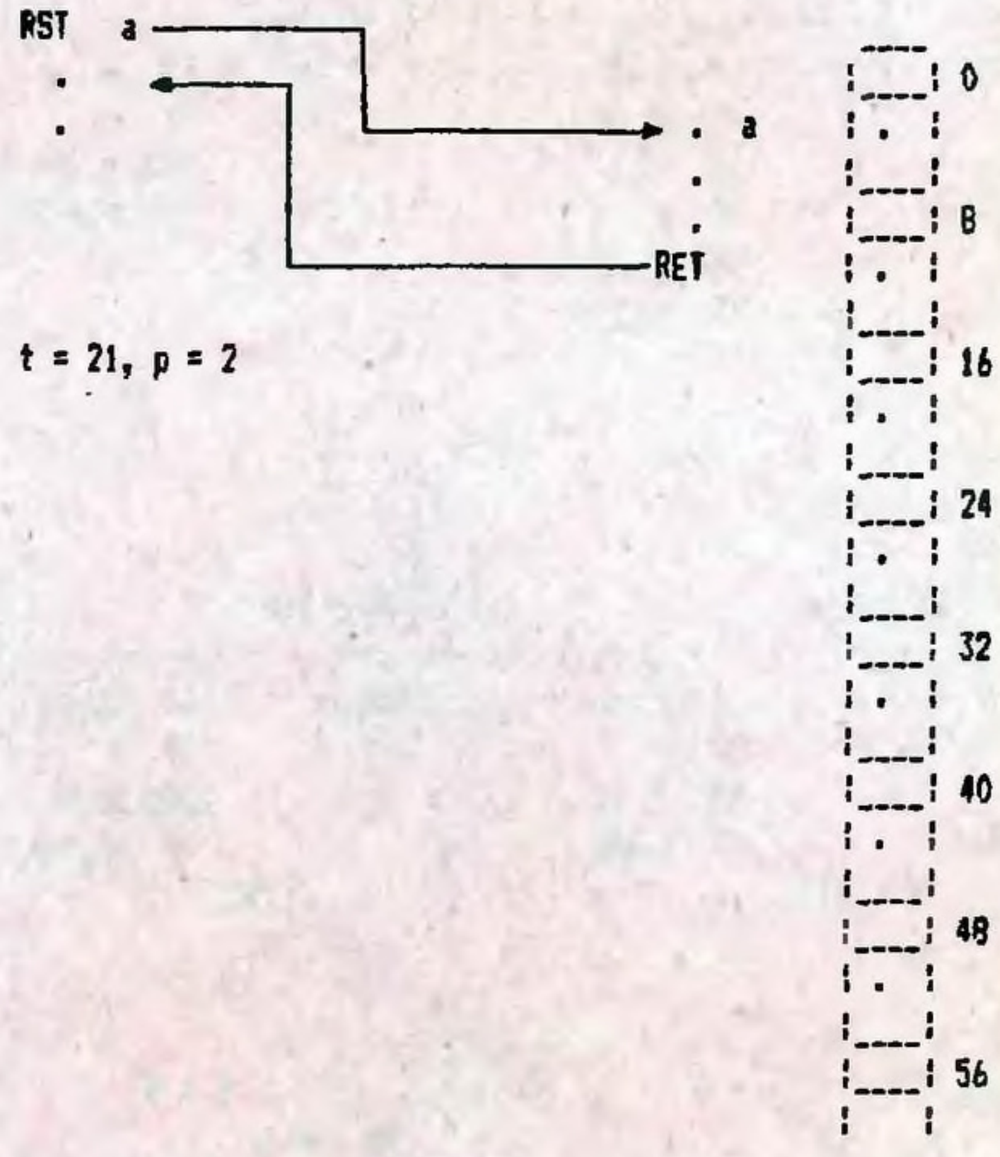
t = 27, p = 4

wywołanie warunkowe i powrót z podprogramu (w = {C, M, NC, NZ, P, PE, PO, Z})



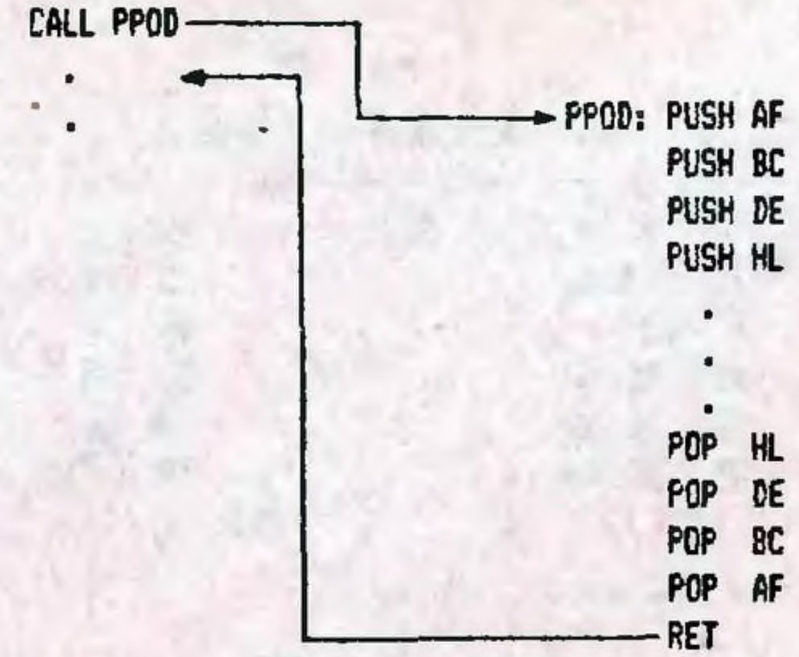
t = 27, p = 4

wywołanie podprogramu przy użyciu rozkazu <RST a>, umieszczonego w pamięci o adresie - a (a = {0, 8, 16, 24, 32, 40, 48, 56})



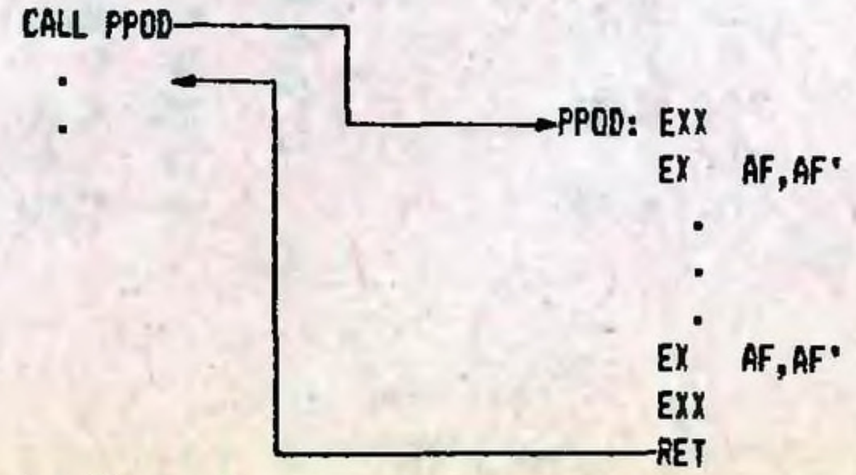
t = 21, p = 2

wywołanie i powrót z podprogramu z zapamiętaniem zawartości rejestrów roboczych na stosie



t = 111, p = 12

wywołanie i powrót z podprogramu z zapamiętaniem zawartości rejestrów roboczych w rejestrach pomocniczych



t = 43, p = 8

4.2. Przekazywanie parametrów do podprogramów

Przedstawione poniżej metody zilustrowano dla przypadku przekazywania parametru jednobajtowego (PAR8) i dwubajtowego (PAR16). Pierwszy przesyłany jest do rejestru A, drugi do pary rejestrów DE.

```
!____!
!____!  PAR8
!____!  PAR16
!____!
!    !
```

przekazywanie parametrów poprzez wspólny obszar pamięci

```
CALL PPOD ---------+
  .    <---+       |
  .        |       +------> PPOD:  .
           |                       .
           |               LD   A,(PAR8)
           |               LD   DE,(PAR16)
           |                       .
           |                       .
           +---------------------- RET
t = 60, p = 11
```

przekazywanie parametrów poprzez podanie adresu ich początku w rejestrach HL

```
LD   HL,PAR8
CALL PPOD ---------+
  .    <---+       |
  .        |       +------> PPOD:  .
           |                       .
           |               LD   A,(HL)
           |               INC  HL
           |               LD   E,(HL)
           |               INC  HL
           |               LD   D,(HL)
           |                       .
           |                       .
t = 70, p = 12 +------------------ RET
```

przekazywanie parametrów poprzez rejestry

```
LD   A,(PAR8)
LD   DE,(PAR16)
CALL PPOD ---------+
  .    <---+       |
  .        |       +------> PPOD:  .
           |                       .
           |                       .
           +---------------------- RET
t = 60, p = 11
```

przekazywanie parametrów za rozkazem CALL przy użyciu pary rejestrów HL

```
CALL PPOD ---------+
DEFB 10            +------> PPOD:  .
DEFW 2345                          .
  .    <---+               POP  HL
  .        |               LD   A,(HL)
           |               INC  HL
           |               LD   E,(HL)
           |               INC  HL
           |               LD   D,(HL)
           |               INC  HL
           |               PUSH HL
           |                       .
           |                       .
t = 104, p = 12 +----------------- RET
```

przekazywanie parametrów za rozkazem CALL przy użyciu rejestru indeksowego IX lub IY (xy)

```
CALL PPOD ---+-----------------------+
DEFB 10      |                       |
DEFW 2345    +--> PPOD:  .           +--> PPOD:  .
  .  <---+           <---------+                 .
  .      |        POP  xy      |           EX   (SP),xy
         |        LD   A,(xy)  |           LD   A,(xy)
         |        LD   E,(xy+1)|           INC  xy
         |        LD   D,(xy+2)|           LD   E,(xy)
         |        LD   BC,3    |           INC  xy
         |        ADD  xy,BC   |           LD   D,(xy)
         |        PUSH xy      |           INC  xy
         |             .       |           EX   (SP),xy
         |             .       |                 .
         |             .       |                 .
         +-------- RET         +---------- RET
     t = 138, p = 22           t = 160, p = 23
```

przekazywanie parametrów poprzez stos

```
LD   BC,(PAR8-1)
PUSH BC
LD   BC,(PAR16)
PUSH BC
CALL PPOD ---------+
  .    <---+       |                .
  .        |       +------> PPOD:   .
           |                        .
           |                POP  BC
           |                POP  DE
           |                POP  AF
           |                PUSH BC
           |                        .
           |                        .
t = 120, p = 17 +------------------ RET
```

5. KONWERSJE POL DANYCH

5.1. Liczby binarne i znakowe w kodzie BCD

konwersja 1-bajtowej liczby binarnej bez znaku (ADR1) na 2-bajtową liczbę w kodzie BCD (ADR2)

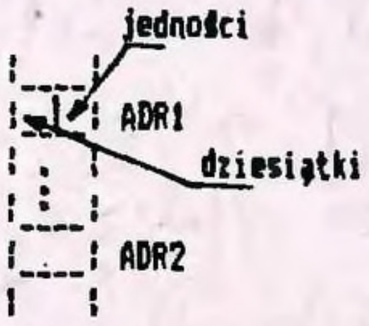
```
!____!
!____! ADR1
! .  !          dziesiątki
! .  !          jednośdci
!____!  ADR2
!____!
!    !          setki
```

```
      LD   A,(ADR1)
      LD   C,-1
P:    INC  C            ;obliczanie
      SUB  100          ;setek
      JR   NC,P
      ADD  A,100
;
      LD   H,A
      LD   A,C
      LD   (ADR2+1),A
      LD   A,H
;
      LD   C,-1
P1:   INC  C            ;obliczanie
      SUB  10           ;dziesiątek
      JR   NC,P1
      ADD  A,10
```

```
        LD   H,A
        LD   A,C
        RLCA                ;przesuwanie dziesiątek
        RLCA                ;na 4 bardziej znaczące
        RLCA                ;pozycje
        RLCA
        DR   H              ;dołączanie jedności
        LD   (ADR2),A
```

t = 23$(s + d) + 143, p = 37
s - liczba setek (0 <= s <= 2)
d - liczba dziesiątek (0 <= d <= 9)

konwersja 1-bajtowej liczby bez znaku w kodzie BCD (ADR1) na 1-bajtową liczby binarną (ADR2)

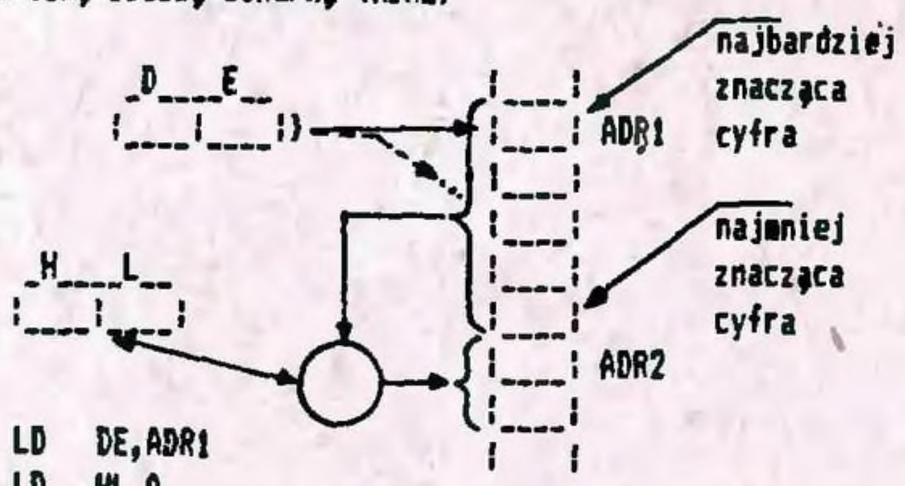



```
        LD   A,(ADR1)
        LD   B,A
        AND  11110000B      ;wycięcie dziesiątek
        RRCA                ;dziesiątki$8
        LD   C,A
        RRCA
        RRCA                ;dziesiątki$2
        ADD  A,C            ;dziesiątki$(B+2)
        LD   C,A
        LD   A,B
        AND  00001111B      ;wycięcie jedności
        ADD  A,C
        LD   (ADR2),A
```

t = 76, p = 19

5.2. Liczby binarne i znakowe w kodzie ASCII

konwersja 5-cio znakowej liczby dziesiętnej w kodzie ASCII (ADR1) na 16-bitową liczbę binarną (ADR2)




```
        LD   DE,ADR1
        LD   HL,0
        LD   B,5
P:      LD   A,(DE)     ;dodanie kolejnej
        SUB  '0'        ;cyfry
        PUSH BC
        LD   C,A
        LD   B,0
        ADD  HL,BC
        INC  DE
        POP  BC
        DEC  B
        JR   Z,P1
        PUSH DE
;                       mnożenie przez 10
        ADD  HL,HL      ; HL$2
        LD   E,L
        LD   D,H
        ADD  HL,HL      ; HL$4
        ADD  HL,HL      ; HL$8
        ADD  HL,DE      ; HL$(8+2)
        POP  DE
        JR   P
P1:     LD   (ADR2),HL
```

t = 755, p = 34

6. OPERACJE NA POLACH BINARNYCH

Operacje na polach binarnych wykonywane są przy użyciu specjalnie do tego celu przeznaczonych rozkazów (SET, RES, BIT), rozkazów logicznych (OR, AND, XOR, CPL), oraz licznej grupy rozkazów przesuwania.

Oznaczenie pozycji binarnych bajtu

6.1. Wprowadzanie określonych wartości na pozycje binarne komórki

wprowadzanie "1" i "0" na b-tą pozycję binarną (b=(0,1,...,7))

```
wprowadzenie "1"          wprowadzenie "0"

LD   HL,ADR               LD   HL,ADR
SET  b,(HL)               RES  b,(HL)

t = 25, p = 5             t = 25, p = 5
```

wprowadzanie "1" na określone pozycje binarne z pozostawieniem bez zmiany pozostałych pozycji (przykładowo wprowadzono "1" na pozycje nr 2, 1 i 0)

```
                          zawartość komórki
LD   A,(ADR)                przed wykonaniem operacji:   01111001
OR   00000111B
LD   (ADR),A                po wykonaniu operacji:       01111111

t = 33, p = 8

LD   HL,ADR
SET  2,(HL)
SET  1,(HL)
SET  0,(HL)

t = 55, p = 9
```

wprowadzanie "0" na określone pozycje binarne z pozostawieniem bez zmiany pozostałych pozycji (przykładowo wprowadzono "0" na pozycje nr 2, 1 i 0)

```
                          zawartość komórki
LD   A,(ADR)                przed wykonaniem operacji:   11101011
AND  11111000B
LD   (ADR)                  po wykonaniu operacji:       11101000

 t = 33, p = 8

LD   HL,ADR
RES  2,(HL)
RES  1,(HL)
RES  0,(HL)

t = 55, p = 9
```

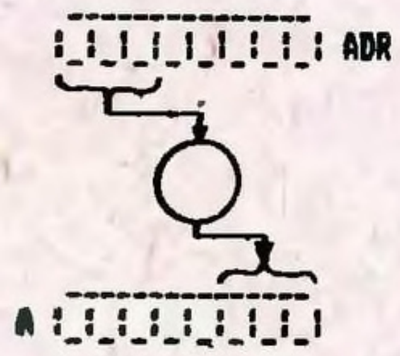
zamiana na wartość przeciwną (z "1" na "0" a z "0" na "1") określonych pozycji binarnych z pozostawieniem bez zmiany pozostałych pozycji (przykładowo zamieniono wartości pozycji nr 2, 1 i 0)

```
                        zawartość komórki
LD   A,(ADR)              przed wykonaniem operacji:    11101011
XOR  00000111B
LD   (ADR),A              po wykonaniu  operacji:       11101100

t = 33, p = 8
```

6.2. Przesłania określonych pozycji binarnych

przesłanie określonych pozycji komórki do rejestru A z prawostronnym przesunięciem (przykładowo przesłano pozycje nr 7, 6 i 5)

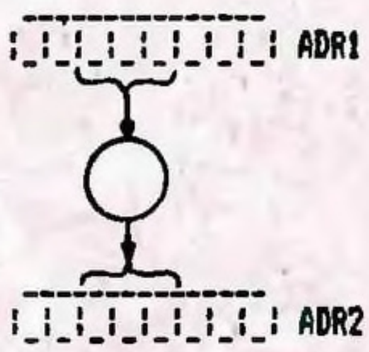
```
LD   A,(ADR)
AND  11100000B
SRL  A
SRL  A
SRL  A
SRL  A
SRL  A
```




```
t = 60, p =15

LD   A,(ADR)
AND  11100000B
RLCA
RLCA
RLCA

t = 32, p = 8
```

przesłanie określonych pozycji binarnych jednej komórki (ADR1) na te same pozycje drugiej komórki (ADR2) z pozostawieniee bez zmiany pozostałych pozycji (przykładowo przesłano pozycje nr 5, 4 i 3)

```
LD   A,(ADR1)
AND  00111000B
LD   B,A
LD   A,(ADR2)
OR   B
LD   (ADR2),A

t = 54, p = 13
```



6.3. Badanie wartości określonych pozycji binarnych

badanie wystąpienia "1" i "0" na określonej pozycji binarnej komórki

(b) = "1"?	(b) = "0"?
LD A,(ADR) BIT b,A JR NZ,TAK NIE: ...	LD A,(ADR) BIT b,A JR Z,TAK NIE: ...
t = {33 - tak, p = 7; 28 - nie}	
LD HL,ADR BIT b,(HL) JR NZ,TAK NIE: ...	LD HL,ADR BIT b,(HL) JR Z,TAK NIE: ...
t = {34 - tak, p = 7; 29 - nie}	

przykładowo badana jest pozycja nr 0	
LD A,(ADR) AND 00000001B JR NZ,TAK NIE: ...	LD A,(ADR) AND 00000001B JR Z,TAK NIE: ...
t = {32 - tak, p = 7; 27 - nie}	

b=(0,1,...,7)

badanie wystąpienia "1" na określonych pozycjach binarnych komórki (przykładowo badane są pozycje nr 2, 1 i 0)

```
      LD   A,(ADR)
      AND  00000111B
      XOR  00000111B
      JR   Z,TAK
NIE:  ...

t = {39 - tak, p = 9; 34 - nie}

      LD   A,(ADR)
      BIT  2,A
      JR   Z,NIE
      BIT  1,A
      JR   Z,NIE
      BIT  0,A
      JR   NZ,TAK
NIE:  ...

t = {63 - tak, p = 15; 33, 48, 58 - nie}
```

badanie wystąpienia "0" na określonych pozycjach binarnych komórki (przykładowo badane są pozycje nr 2, 1 i 0)

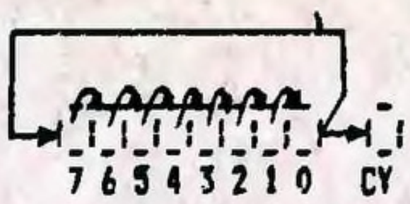
```
      LD   A,(ADR)
      AND  00000111B
      JR   Z,TAK
NIE:  ...

t = {32 - tak, p = 7; 27 - nie}

      LD   A,(ADR)
      BIT  2,A
      JR   NZ,NIE
      BIT  1,A
      JR   NZ,NIE
      BIT  0,A
      JR   Z,TAK
NIE:  ...

t = {63 - tak, p = 15; 33, 48, 58 - nie}
```

6.4. Przesuwanie pozycji binarnych

Oznaczenia:
CY - wskaźnik przeniesienia
bzb - bardziej znaczący bajt
mzb - mniej znaczący bajt

6.4.1. Przesuwanie cykliczne w lewo i w prawo (o jedną pozycję) wielkości 8-bitowej

przesuwanie w lewo



przesuwanie w prawo

przesuwanie zawartości akumulatora

```
RLCA                                    RRCA
                t = 4, p = 1
```

przesuwanie zawartości rejestru A, B, C, D, E, H lub L (r)

```
RLC  r                                  RRC  r
                t = 8, p = 2
```

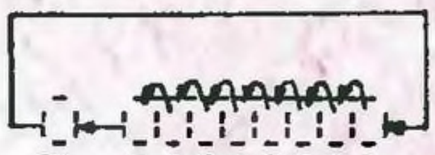
przesuwanie zawartości komórki (ADR) przy użyciu pary rejestrów HL

```
LD   HL,ADR                             LD   HL,ADR
RLC  (HL)                               RRC  (HL)
                t = 25, p = 5
```

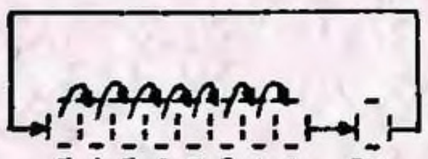
przesuwanie zawartości komórki (ADR) przy użyciu rejestru IX lub IY (xy)

```
LD   xy,ADR                             LD   xy,ADR
RLC  (xy)                               RRC  (xy)
                t = 37, p = 8
```

6.4.2. Przesuwanie cykliczne w lewo i w prawo z przeniesieniem (o jedną pozycję) wielkości 8-bitowej



przesuwanie w lewo



przesuwanie w prawo

przesuwanie zawartości akumulatora

```
RLA                                     RRA
                t = 4, p = 1
```

przesuwanie zawartości rejestru A, B, C, D, E, H lub L (r)

```
RL   r                                  RR   r
                t = 8, p = 2
```

przesuwanie zawartości komórki (ADR) przy użyciu pary rejestrów HL

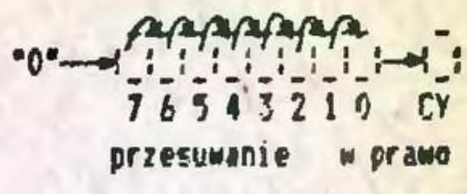
```
LD   HL,ADR                             LD   HL,ADR
RL   (HL)                               RR   (HL)
                t = 25, p = 5
```

przesuwanie zawartości komórki (ADR) przy użyciu rejestru IX lub IY (xy)

```
LD   xy,ADR                             LD   xy,ADR
RL   (xy)                               RR   (xy)
                t = 37, p = 8
```

6.4.3. Przesuwanie logiczne w lewo i w prawo (o jedną pozycję) wielkości 8-bitowej

przesuwanie w lewo



przesuwanie w prawo

przesuwanie zawartości akumulatora

```
ADD  A
```

t = 4, p = 1

przesuwanie zawartości rejestru A, B, C, D, E, H lub L (r)

```
SLA  r                                  SRL  r
                t = 8, p = 2
```

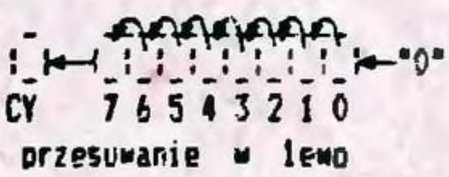
przesuwanie zawartości komórki (ADR) przy użyciu pary rejestrów HL

```
LD   HL,ADR                             LD   HL,ADR
SLA  (HL)                               SRL  (HL)
                t = 25, p = 5
```

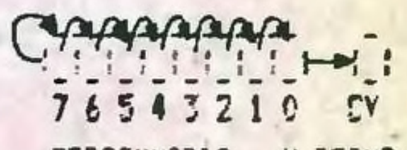
przesuwanie zawartości komórki (ADR) przy użyciu rejestru IX lub IY (xy)

```
LD   xy,ADR                             LD   xy,ADR
SLA  (xy)                               SRL  (xy)
                t = 37, p = 9
```

6.4.4. Przesuwanie arytmetyczne w lewo i w prawo (o jedną pozycję) wielkości 8-bitowej



przesuwanie w lewo



przesuwanie w prawo

przesuwanie zawartości rejestru A, B, C, D, E, H lub L (r)

```
SLA  r                                  SRA  r
                t = 8, p = 2
```

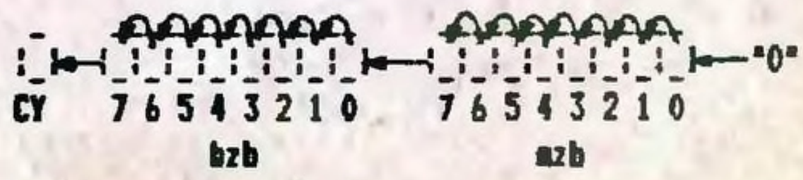
przesuwanie zawartości komórki (ADR) przy użyciu pary rejestrów HL

```
LD   HL,ADR                             LD   HL,ADR
SLA  (HL)                               SRA  (HL)
                t = 25, p = 5
```

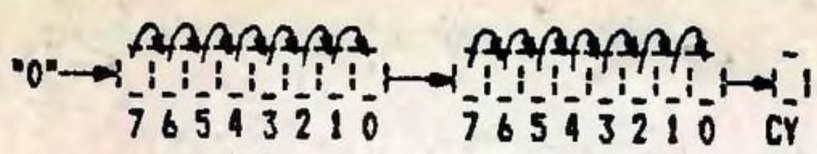
przesuwanie zawartości komórki (ADR) przy użyciu rejestru IX lub IY (xy)

```
LD   xy,ADR                             LD   xy,ADR
SLA  (xy)                               SRA  (xy)
                t = 37, p = 8
```

6.4.5. Przesuwanie logiczne w lewo i w prawo (o jedną pozycję) wielkości 16-bitowej



przesuwanie w lewo



przesuwanie w prawo

przesuwanie zawartości rejestrów HL

```
ADD  HL,HL                    SRL  H
                              RR   L
t = 11, p = 1                 t = 16, 4
```

przesuwanie zawartości rejestru IX lub IY (xy)

```
ADD xy,xy

t = 15 p = 2
```

przesuwanie zawartości pary rejestrów BC, DE, HL (rb = (B, D, H), rm = (C, E, L))

```
SLA  rm                       SRL  rb
RL   rb                       RR   rm
                t = 16, p = 4
```

przesuwanie zawartości 2 komórek (ADR i ADR+1) przy użyciu pary rejestrów HL i rejestru IX lub IY (xy)

```
LD   HL,ADR                   LD   ADR+1
SLA  (HL)                     SRL  (HL)
INC  HL                       DEC  HL
RL   (HL)                     RR   (HL)
                t = 46, p = 8

LD   xy,ADR                   LD   xy,ADR
SLA  (xy)                     SRL  (xy+1)
RL   (xy+1)                   RR   (xy)
                t = 60, p = 12
```

7. OPERACJE ARYTMETYCZNE

Ze względu na brak w mikroprocesorze Z80 rozkazów mnożenia i dzielenia, operacje te wykonywane są przy użyciu rozkazów dodawania, odejmowania i przesuwania.

7.1. Dodawanie

7.1.1. Dodawanie liczb binarnych

dodawanie liczb 1-bajtowych

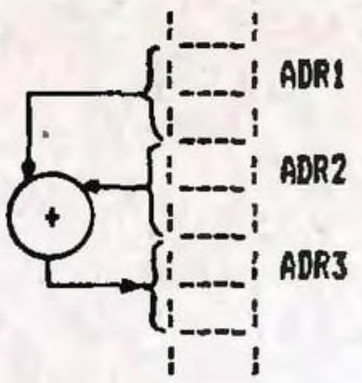
```
LD   A,(ADR1)  ;ADR3:=ADR1+ADR2
LD   HL,ADR2
ADD  A,(HL)
LD   (ADR3),A

t = 43, p = 10
```



```
LD   A,(ADR1)  ;wersja bez użycia
LD   B,A       ;rejestrów HL
LD   A,(ADR2)
ADD  A,B
LD   (ADR3),A

t = 47, P = 11
```

dodawanie liczb 2-bajtowych

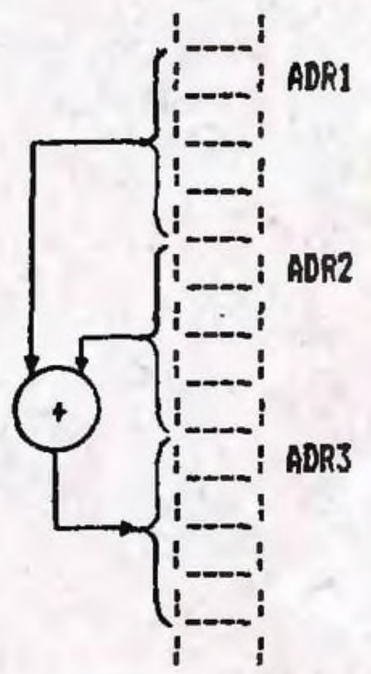
```
LD   A,(ADR1)  ;ADR3:=ADR1+ADR2
LD   HL,ADR2
ADD  A,(HL)
LD   (ADR3),A
LD   A,(ADR1+1)
DEC  HL
ADC  A,(HL)
LD   (ADR3+1),A

t = 82, p = 18
```




```
LD   HL,(ADR1) ;ADR3:=ADR1+ADR2
LD   BC,(ADR2)
ADD  HL,BC
LD   (ADR3),HL

t = 63, p = 11
```

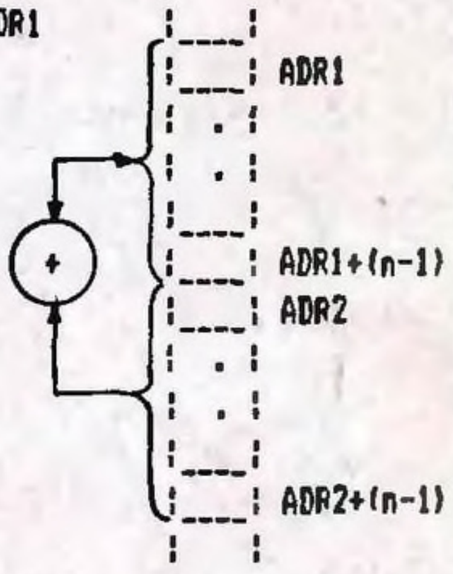
dodawanie liczb 4-bajtowych

```
LD   HL,(ADR1) ;ADR3:=ADR1+ADR2
LD   BC,(ADR2)
ADD  HL,BC
LD   (ADR3),HL
LD   HL,(ADR1+2)
LD   BC,(ADR2+2)
ADC  HL,BC
LD   (ADR3+2),HL

t = 130, p = 23
```



dodawanie liczb n-bajtowych

```
     LD   HL,ADR1  ;ADR1:=ADR2+ADR1
     LD   DE,ADR2
     LD   B,n
     AND  A
P:   LD   A,(DE)
     ADC  A,(HL)
     LD   (HL),A
     INC  HL
     INC  DE
     DJNZ P

t = 46*n + 29, p = 17
(1 <= n <= 255)
```



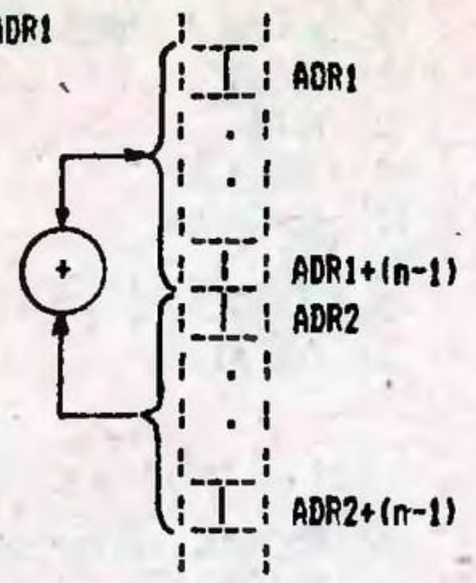
7.1.2. Dodawanie liczb w kodzie BCD

dodawanie liczb 1-bajtowych bez znaku

```
LD   A,(ADR1)  ;ADR3:=ADR1+ADR2
LD   HL,ADR2
ADD  A,(HL)
DAA
LD   (ADR3),A

t = 47,  p = 11
```

dodawanie liczb n-bajtowych bez znaku

```
     LD   HL,ADR1  ;ADR1:=ADR2+ADR1
     LD   DE,ADR2
     LD   B,n
     AND  A
P:   LD   A,(DE)
     ADC  A,(HL)
     DAA
     LD   (HL),A
     INC  HL
     INC  DE
     DJNZ P
```



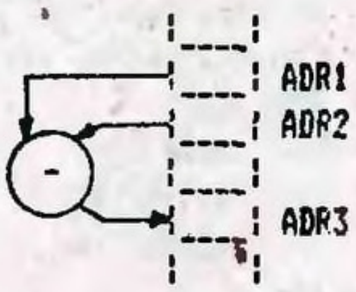
t = 568n + 29, p = 18
(1 <= n <= 255)

7.2. Odejmowanie

7.2.1. Odejmowanie liczb binarnych

odejmowanie liczb 1-bajtowych

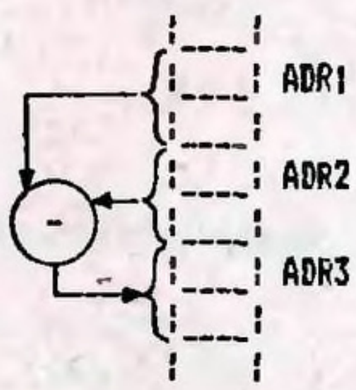
```
LD   A,(ADR1)  ;ADR3:=ADR1-ADR2
LD   HL,ADR2
SUB  A,(HL)
LD   (ADR3),A
```



t = 43, p = 10

odejmowanie liczb 2-bajtowych

```
LD   HL,(ADR1) ;ADR3:=ADR1-ADR2
LD   DE,ADR2
AND  A
SBC  HL,DE
LD   (ADR3),HL
```

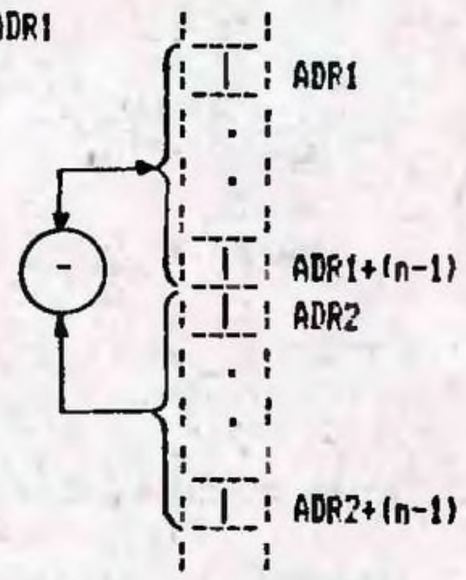


t = 82, p = 18

7.2.2. Odejmowanie liczb w kodzie BCD

odejmowanie liczb n-bajtowych

```
     LD   HL,ADR1  ;ADR1:=ADR2-ADR1
     LD   DE,ADR2
     LD   B,n
     AND  A
P:   LD   A,(DE)
     SBC  A,(HL)
     DAA
     LD   (HL),A
     INC  HL
     INC  DE
     DJNZ P
```

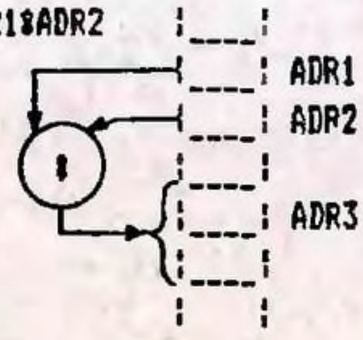


t = 56*n + 29, p = 18
(1 <= n <= 255)

7.3 Mnożenie liczb binarnych

mnożenie liczb 1-bajtowych bez znaku (wynik 2-bajtowy)

```
     LD   HL,(ADR1-1) ;ADR3:=ADR1*ADR2
     LD   L,0
     LD   D,0
     LD   A,(ADR2)
     LD   E,A
     LD   B,8
P:   ADD  HL,HL
     JR   NC,P1
     ADD  HL,DE
P1:  DJNZ P
     LD   (ADR3),HL
```



t ≈ 377, p = 22

mnożenie zawartości rejestru A przez 2

```
ADD  A,A     ;A:=A*2
```

t = 4, p = 1

```
SLA  A       ;A:=A*2
```

t = 8 ,p = 2

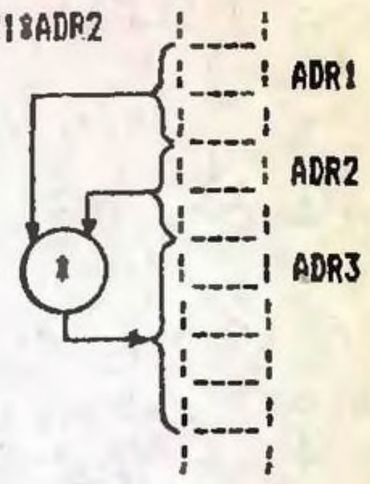
mnożenie zawartości rejestru A przez 5

```
LD   B,A     ;A*(2+2+1)
ADD  A,A
ADD  A,A
ADD  A,B
```

t = 16, p = 4

mnożenie liczb 2-bajtowych bez znaku (wynik 4-bajtowy)

```
     LD   BC,(ADR1)   ;ADR3:=ADR1*ADR2
     LD   DE,(ADR2)
     LD   HL,0
     LD   A,16
P:   ADD  HL,HL
     EX   DE,HL
     ADC  HL,HL
     JR   NC,P1
     ADD  HL,BC
     JR   NC,P1
     INC  DE
P1:  DEC  A
     JR   NZ,P
     LD   (ADR3),HL
     LD   (ADR3+2),DE
```



t ≈ 1208, p = 33

mnożenie liczby 2-bajtowej przez 2

```
LD   HL,ADR   ;ADR:=ADR*2
SLA  (HL)
INC  HL
RL   (HL)
```

t = 46, p = 8

```
LD   xy,ADR   ;ADR:=ADR*2
SLA  (xy)
RL   (xy+1)
```

t = 60, p = 12

mnożenie liczby 2-bajtowej przez 10

```
LD   HL,(ADR) ;ADR:=ADR*10
ADD  HL,HL    ;HL*2
LD   E,L
LD   D,H
ADD  HL,HL    ;HL*4
ADD  HL,HL    ;HL*8
ADD  HL,DE    ;HL*(8+2)
LD   (ADR),HL
```

t = 84, p = 12

7.4 Dzielenie liczb binarnych

dzielenie liczby bez znaku w rejestrze A przez 2

```
SRL  A       ;A:=A/2
```

t = 8, p = 2

dzielenie liczby ze znakiem w rejestrze A przez 2

```
SRA  A        ;A1=A/2
```

t = 8, p = 2

dzielenie liczby 2-bajtowej bez znaku przez 2

```
LD   xy,ADR   ;ADR1=ADR/2
SRL  (xy+1)
RR   (xy)
```

t = 60, p = 12

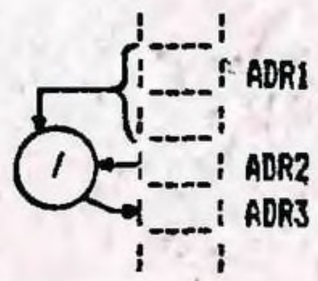
dzielenie liczby 2-bajtowej ze znakiem przez 2

```
LD   xy,ADR   ;ADR1=ADR/2
SRA  (xy+1)
RR   (xy)
```

t = 60, p = 12

dzielenie liczby 2-bajtowej przez liczbę 1-bajtową

```
     LD   HL,(ADR1)   ;ADR3=ADR1/ADR2
     LD   A,(ADR2)
     LD   C,A
     LD   B,8
P:   ADD  HL,HL
     LD   A,H
     SUB  C
     JR   C,P1
     LD   H,A
     INC  L
P1:  DJNZ P
     LD   A,L
     LD   (ADR3),A
```



reszta z dzielenia w rejestrze H

t ≅ 452, p = 16

8. OPERACJE NA TABLICACH

Oznaczenia:
n - liczba elementów tablicy
k - liczba porównań w tablicy

8.1. Sekwencyjne przeglądanie tablicy

8.1.1. Przeglądanie tablicy zbudowanej z elementów 1-bajtowych

Przeglądanie przy użyciu rozkazu CPIR

```
     LD   A,(SZUKAJ)
     LD   HL,ADR
     LD   BC,n
     CPIR
     JR   Z,JEST
```

brak szukanego elementu

JEST:

w HL adres szukanego elementu

t = 21*k + 40 - znaleziono szukany element
t = 21*n + 49 - brak szukanego elementu
, p = 13
(1 <= k <= n)
(1 <= n <= 65535)

Dla n-elementowej tablicy liczba porównań k zależy od aktualnego położenia szukanego elementu. W najbardziej korzystnym przypadku k = 1, a najbardziej niekorzystnym k = n. Zakładając, że położenie poszukiwanego elementu w tablicy podlega równomiernemu rozkładowi prawdopodobieństwa, średnio należy przeprowadzić (n + 1)/2 porównań.

Przeglądanie przy użyciu rozkazu CPI

```
     LD   A,(SZUKAJ)
     LD   HL,ADR
     LD   BC,n
P:   CPI
     JR   Z,JEST
     JP   PE,P
```

BRAK:

brak szukanego elementu

JEST:

w HL adres szukanego elementu

t = 33*k + 28 - znaleziono szukany element
t = 33*n + 28 - brak szukanego elementu
, p = 16
(1 <= k <= n)
(1 <= n <= 65535)

Przeglądanie przy użyciu rozkazu CP (HL)

```
     LD   A,(SZUKAJ)
     LD   HL,ADR
     LD   B,n
P:   CP   (HL)
     JR   Z,JEST
     INC  HL
     DJNZ P
```

brak szukanego elementu

JEST:

w HL adres szukanego elementu

t = 33*k + 16 - znaleziono szukany element
t = 33*n + 25 - brak szukanego elementu
, p = 13
(1 <= k <= n)
(1 <= n <= 255)

8.1.2. Przeglądanie tablicy zbudowanej z elementów 2-bajtowych

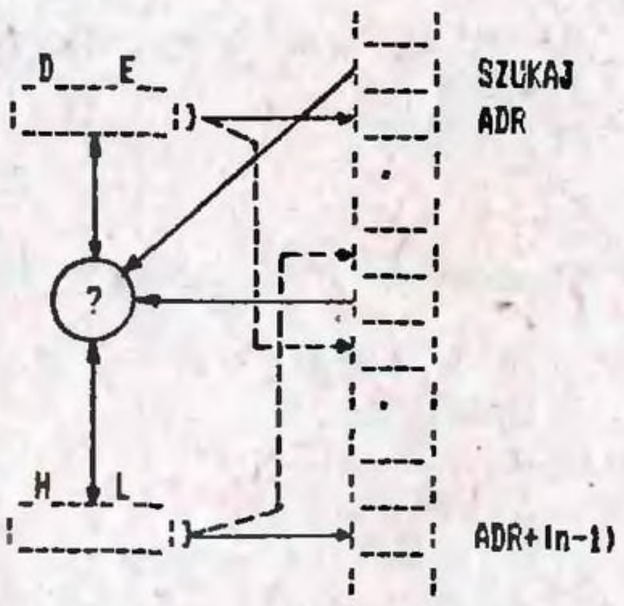
```
     LD   A,(SZUKAJ)
     LD   HL,ADR
     LD   B,n
P1:  CP   (HL)
     JR   Z,P2
     INC  HL
     INC  HL
     DJNZ P1
     JR   BRAK
P2:  LD   A,(SZUKAJ+1)
     INC  HL
     LD   D,(HL)
     CP   D
     JR   Z,JEST
     INC  HL
     LD   A,(SZUKAJ)
     DJNZ P1
```

BRAK:
brak szukanego elementu
JEST:
w HL adres do drugiego bajtu szukanego elementu

t = 398k + 498a + 15, p = 31
(1 <= n <= 255)
a - liczba elementów tablicy, których pierwszy bajt jest identyczny z pierwszym bajtem poszukiwanego elementu, a drugie są różne (0 <= a <= n)

8.2. Binarne (logarytmiczne) przeglądanie tablicy zbudowanej z elementów 1-bajtowych

Warunkiem stosowania przeglądania binarnego jest uporządkowanie elementów tablicy wg wartości wzrastających. Metoda ta polega na porównywaniu poszukiwanej wartości ze środkowym elementem tablicy i jeżeli nie są to te same wartości, podobszar dalszego przeszukiwania jest ograniczony do górnej lub dolnej połowy obszaru tablicy, wyznaczonej przez element środkowy. Z kolei w połowie tej badany jest jej element środkowy i znowu, jeżeli warunek porównywania nie jest spełniony, to podobszar przeszukiwania zmniejsza się o połowę. Operacja ta jest powtarzana, aż do znalezienia poszukiwanego elementu lub stwierdzenia jego braku w tablicy.




```
        LD   HL,ADR
        LD   E,L        ;DE:=ADR
        LD   D,H
        LD   BC,n-1
        ADD  HL,BC      ;HL:=ADR+(n-1)
        LD   A,(SZUKAJ)
        LD   C,A
P:      PUSH HL
        ADD  HL,DE      ;HL:=(HL+DE)/2
        RR   H          ;dzielenie 17-bitowej
        RR   L          ;liczby przez 2
        LD   A,(HL)     ;sprawdź element środkowy
        CP   C
        JR   NC,P1
        EX   DE,HL      ;dolna podtablica
        INC  DE
        POP  HL
        JR   P2
P1:     INC  SP
        INC  SP
        JR   Z,JEST
        DEC  HL         ;górna podtablica
P2:     LD   A,L        ;DE > HL gdy brak elementu
        CP   E
        LD   A,H
        SBC  A,D
        JR   NC,P
```

brak szukanego elementu

JEST:

w HL adres szukanego elementu

t = 113tk + 23, p = 41

Maksymalna liczba porównań w tablicy wynosi:

$$k = \lfloor \log_2 n \rfloor + 1$$

Gdy brak poszukiwanego elementu:

$$k = \lfloor \log_2 n \rfloor + 2$$

Zakładając, że położenie poszukiwanego elementu w tablicy podlega równomiernemu rozkładowi prawdopodobieństwa, średnia liczba porównań wynosi [6]:

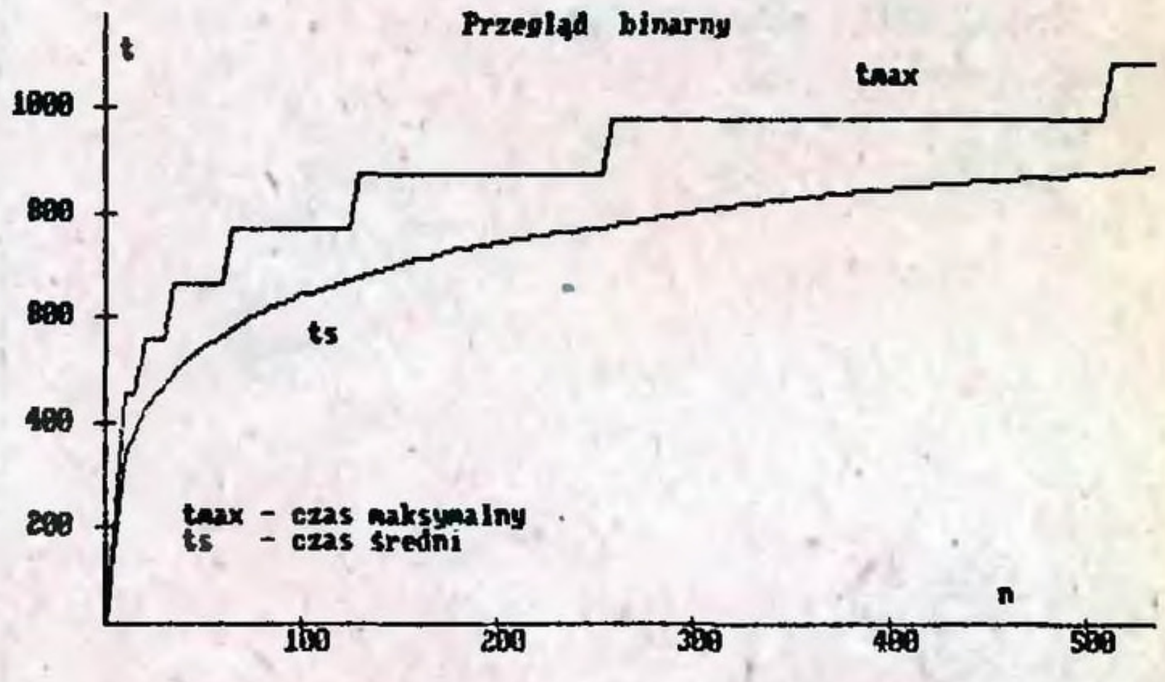
$$k = q + 1 - (2^{q+1} - q - 2)/n$$

gdzie:

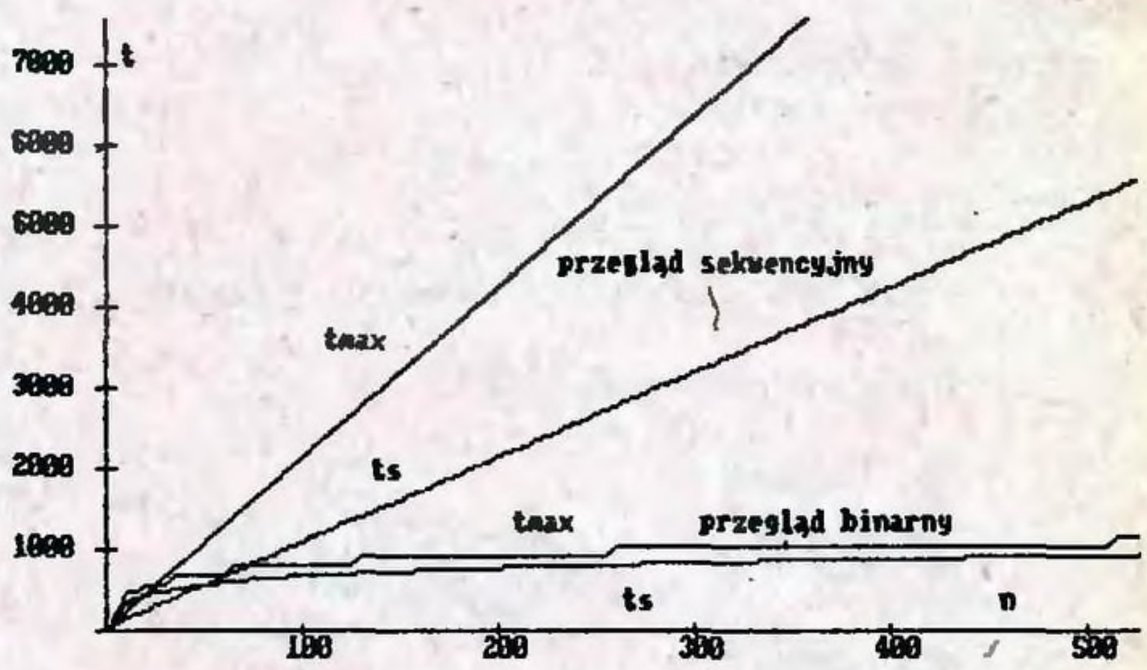
$$q = \lfloor \log_2 n \rfloor$$

$\lfloor r \rfloor$ - oznaczenie części całkowitej liczby r

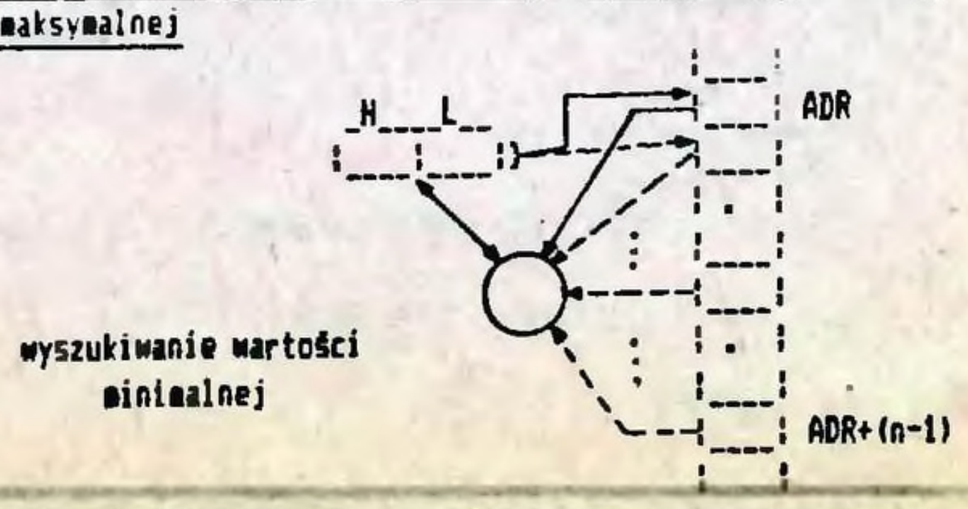
Poniżej zestawiono wykresy ilustrujące zależność średniego i maksymalnego czasu przeglądu binarnego od rozmiarów tablicy. Cechą charakterystyczną jest stosunkowo mała różnica pomiędzy czasem maksymalnym i średnim.



Kolejne wykresy przedstawiają zestawienie charakterystyk chronometrażowych przeglądu sekwencyjnego (przy użyciu rozkazu CPIR) i binarnego. Wykazują one, że dla tablic o niewielkich rozmiarach (do 50 elementów) lepiej jest stosować przegląd sekwencyjny, oraz wzrost przewagi przeglądu binarnego ze wzrostem rozmiarów tablicy.



8.3. Wyszukiwanie z tablicy elementu o wartości minimalnej i maksymalnej



```
      LD   HL,ADR
      LD   B,n
P:    LD   A,(HL)
      LD   E,L
      LD   D,H
P1:   DEC  B
      JR   Z,P2
      INC  HL
      CP   (HL)
      JR   C,P1   ;A < (HL)
      JR   Z,P1   ;A = (HL)
      JR   P      ;A > (HL)
P2:   EX   DE,HL
```

w HL adres elementu o wartości minimalnej

t = 39$n + 26$a2 + 52, p = 20

a - liczba przypadków, gdy kolejny element tablicy ma wartość mniejszą od wartości najmniejszej z dotychczas znalezionych (1 <= a <= n), (2 <= n <= 255)

wyszukiwanie wartości maksymalnej

```
      LD   HL,ADR
      LD   B,n
P:    LD   A,(HL)
      LD   E,L
      LD   D,H
P1:   DEC  B
      JR   Z,P2
      INC  HL
      CP   (HL)
      JR   NC,P1  ;A >= (HL)
      JR   P      ;A <  (HL)
P2:   EX   DE,HL
```

w HL adres elementu o wartości maksymalnej

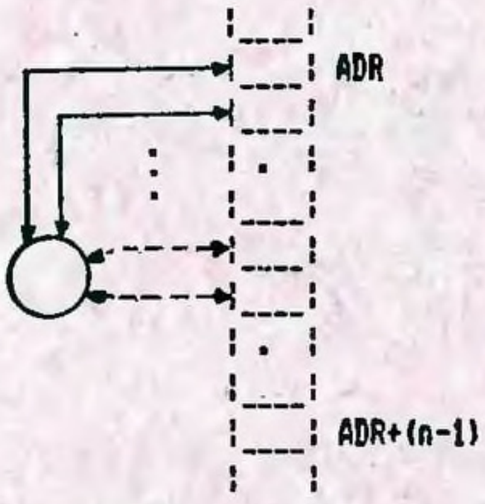
t = 36$n + 22$a + 40, p = 18

a - liczba przypadków, gdy kolejny element tablicy ma wartość większą od wartości największej z dotychczas znalezionych (1 <= a <= n), (2 <= n <= 255)

8.4. Sortowanie elementów tablicy

Sortowanie tablicy (zbudowanej z elementów 1-bajtowych) metodą zamiany sąsiednich elementów.

Metoda ta polega na zasadzie porównywania i zamiany par sąsiadujących ze sobą elementów tablicy tak długo, aż wszystkie zostaną posortowane. Na początku pierwszy element tablicy jest porównywany z drugim i zamieniany z nim w razie potrzeby. Potem drugi z trzecim, trzeci z czwartym i tak aż do końca tablicy. W razie wystąpienia co najmniej jednej zamiany, cała ta operacja jest powtarzana od początku tablicy. Brak zamiany oznacza koniec sortowania.




```
P:    LD   HL,ADR
      LD   DE,ADR+1
      LD   B,n-1
      LD   C,0
;
P1:   LD   A,(DE)
      CP   (HL)
      JR   NC,P2
;
      LD   C,(HL)
      LD   (HL),A
      LD   A,C
```

;porównanie 2 kolejnych ;elementów

;zamiana sąsiednich ;elementów

```
      LD   (DE),A
      LD   C,1          ;była zamiana
P2:   INC  HL
      INC  DE
      DJNZ P1
;
      DEC  C            ;czy była zamiana?
      JR   Z,P
```

p = 27, (2 <= n <= 255)

Czas wykonania operacji sortowania w znacznym stopniu zależy od początkowego ułożenia elementów tablicy. W najbardziej korzystnym wypadku, gdy przy pierwszym przeglądnięciu tablicy nie wystąpiła żadna zamiana (czyli elementy tablicy były już posortowane) czas ten wynosi:

t = n$46 - 6

W najbardziej niekorzystnym przypadku, gdy elementy tablicy są uporządkowane odwrotnie, tzn. od wartości największej do najmniejszej czas sortowania wynosi:

$$t = n\$52 + n\$26$$

DODATEK I.

ZESTAWIENIE ROZKAZÓW MIKROPROCESORA Z80

	t	p	2 MHz	3.5 MHz	4 MHz	
ADC A,n	7	2	3.5	2	1.8	n - wielkość 8-bitowa
ADC A,r	4	1	2	1.1	1	r=(A,B,C,D,E,H,L)
ADC A,(HL)	7	1	3.5	2	1.8	
ADC A,(xy+d)	19	3	9.5	5.4	4.8	xy=(IX,IY)
						d - liczba 8-bitowa ze znakiem
ADC HL,rp	15	2	7.5	4.3	3.8	rp=(BC,DE,HL,SP)
ADD A,n	7	2	3.5	2	1.8	
ADD A,r	4	1	2	1.1	1	
ADD A,(HL)	7	1	3.5	2	1.8	
ADD A,(xy+d)	19	3	9.5	5.4	4.8	
ADD HL,rp	11	1	5.5	3.1	2.8	rp=(BC,DE,HL,SP)
ADD IX,rp	15	2	7.5	4.3	3.8	rp=(BC,DE,IX,SP)
ADD IY,rp	15	2	7.5	4.3	3.8	rp=(BC,DE,IY,SP)
AND n	7	2	3.5	2	1.8	
AND r	4	1	2	1.1	1	
AND (HL)	7	1	3.5	2	1.8	
AND (xy+d)	19	3	9.5	5.4	4.8	
BIT b,r	8	2	4	2.3	2	b=(0,1,...,7)
BIT b,(HL)	12	2	6	3.4	3	
BIT b,(xy+d)	20	4	10	5.7	5	
CALL nn	17	3	8.5	4.9	4.3	nn -wielkość 16-bitowa
CALL C,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL M,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL NC,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	nie-(10/17)-tak
CALL NZ,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL P,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL PE,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL PO,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CALL Z,nn	10/17	3	5/8.5	2.9/4.9	2.5/4.3	
CCF	4	1	2	1.1	1	
CP n	7	2	3.5	2	1.8	
CP r	4	1	2	1.1	1	
CP (HL)	7	1	3.5	2	1.8	
CP (xy+d)	19	3	9.5	5.4	4.8	
CPD	16	2	8	4.6	4	BC=0 OR A=(HL)
CPDR	21/16	2	10.5/8	6/4.6	5.3/4	
CPI	16	2	8	4.6	4	(21/16)
CPIR	21/16	2	10.5/8	6/4.6	5.3/4	BC#0 AND A#(HL)
CPL	4	1	2	1.1	1	

		t	p	2 MHz	3.5 MHz	4 MHz	
DAA		4	1	2	1.1	1	
DEC	r	4	1	2	1.1	1	
DEC	(HL)	11	1	5.5	3.1	2.8	
DEC	(xy+d)	23	3	11.5	6.6	5.8	
DEC	rp	6	1	3	1.7	1.5	rp=(BC,DE,HL,SP)
DEC	xy	10	2	5	2.9	2.5	
DI		4	1	2	1.1	1	
DJNZ	d	13/8	2	6.5/4	3.7/2.3	3.3/2	B#0-(13/8)-B=0
EI		4	1	2	1.1	1	
EX	AF,AF'	4	1	2	1.1	1	
EX	DE,HL	4	1	2	1.1	1	
EX	(SP),HL	19	1	9.5	5.4	4.8	
EX	(SP),xy	23	2	11.5	6.6	5.8	
EXX		4	1	2	1.1	1	
HALT		4	1	2	1.1	1	
IM	0	8	2	4	2.3	2	
IM	1	8	2	4	2.3	2	
IM	2	8	2	4	2.3	2	
IN	A,(n)	11	1	5.5	3.1	2.8	
IN	r,(C)	12	2	6	3.4	3	
INC	r	4	1	2	1.1	1	
INC	(HL)	11	1	5.5	3.1	2.8	
INC	(xy+d)	23	3	11.5	6.6	5.8	
INC	rp	6	1	3	1.7	1.5	rp=(BC,DE,HL,SP)
INC	xy	10	2	5	2.9	2.5	
IND		15	2	7.5	4.3	3.8	
INDR		20/15	2	10/7.5	5.7/4.3	5/3.8	B#0-(20/15)-B=0
INI		15	2	7.5	4.3	3.8	
INIR		20/15	2	10/7.5	5.7/4.3	5/3.8	
JP	nn	10	3	5	2.9	2.5	
JP	(HL)	4	1	2	1.1	1	
JP	(xy)	8	2	4	2.3	2	
JP	C,nn	10	3	5	2.9	2.5	
JP	M,nn	10	3	5	2.9	2.5	
JP	NC,nn	10	3	5	2.9	2.5	
JP	NZ,nn	10	3	5	2.9	2.5	
JP	P,nn	10	3	5	2.9	2.5	
JP	PE,nn	10	3	5	2.9	2.5	
JP	PO,nn	10	3	5	2.9	2.5	
JP	Z,nn	10	3	5	2.9	2.5	

		t	p	2 MHz	3.5 MHz	4 MHz	
JR	d	12	2	6	3.4	3	
JR	C,d	7/12	2	3.5/6	2/3.4	1.8/3	nie-(7/12)-tak
JR	NC,d	7/12	2	3.5/6	2/3.4	1.8/3	
JR	NZ,d	7/12	2	3.5/6	2/3.4	1.8/3	
JR	Z,d	7/12	2	3.5/6	2/3.4	1.8/3	
LD	r1,r2	4	1	2	1.1	1	r1=r2=(A,B,C,D,E,H,L)
LD	r,n	7	2	3.5	2	1.8	
LD	r,(HL)	7	1	3.5	2	1.8	
LD	r,(xy+d)	19	3	9.5	5.4	4.8	
LD	(HL),n	10	2	5	2.9	2.5	
LD	(HL),r	7	1	3.5	2	1.8	
LD	(xy+d),r	19	3	9.5	5.4	4.8	
LD	(xy+d),n	19	4	9.5	5.4	4.8	
LD	A,(rp)	7	1	3.5	2	1.8	rp=(BC,DE)
LD	A,(nn)	13	3	6.5	3.7	3.3	
LD	(rp),A	7	1	3.5	2	1.8	rp=(BC,DE)
LD	(nn),A	13	3	6.5	3.7	3.3	
LD	A,I	9	2	4.5	2.6	2.3	
LD	A,R	9	2	4.5	2.6	2.3	
LD	I,A	9	2	4.5	2.6	2.3	
LD	R,A	9	2	4.5	2.6	2.3	
LD	rp,nn	10	3	5	2.9	2.5	rp=(BC,DE,HL,SP)
LD	xy,nn	14	4	7	4	3.5	
LD	HL,(nn)	16	3	8	4.6	4	
LD	rp,(nn)	20	4	10	5.7	5	rp=(BC,DE,SP,IX,IY)
LD	(nn),HL	16	3	8	4.6	4	
LD	(nn),rp	20	4	10	5.7	5	rp=(BC,DE,SP,IX,IY)
LD	SP,HL	6	1	3	1.7	1.5	
LD	SP,xy	10	2	5	2.9	2.5	
LDD		16	2	8	4.6	4	
LDDR		21/16	2	10.5/8	6/4.6	5.3/4	BC#0-(21/16)-BC=0
LDI		16	2	8	4.6	4	
LDIR		21/16	2	10.5/8	6/4.6	5.3/4	
NEG		8	2	4	2.3	2	
NOP		4	1	2	1.1	1	
OR	n	7	2	3.5	2	1.8	
OR	r	4	1	2	1.1	1	
OR	(HL)	7	1	3.5	2	1.8	
OR	(xy+d)	19	3	9.5	5.4	4.8	
OUT	(C),r	12	2	6	3.4	3	
OUT	(n),A	11	2	5.5	3.1	2.8	
OUTI		16	2	8	4.6	4	
OTIR		21/16	2	10.5/8	6/4.6	5.3/4	B#0-(21/16)-B=0
OUTD		16	2	8	4.6	4	
OTDR		21/16	2	10.5/8	6/4.6	5.3/4	

		t	p	2 MHz	3.5 MHz	4 MHz	
POP	rp	10	1	5	2.9	2.5	rp=(BC.DE.HL.AF)
POP	xy	14	2	7	4	3.5	
PUSH	rp	11	1	5.5	3.1	2.8	rp=(BC,DE,HL.AF)
PUSH	xy	15	2	7.5	4.3	3.8	
RES	b,r	8	2	4	2.3	2	
RES	b,(HL)	15	2	7.5	4.3	3.8	
RES	b,(xy+d)	23	4	11.5	6.6	5.8	
RET		10	1	5	2.9	2.5	
RET	C	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	nie-(5/11)-tak
RET	M	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET	NC	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET	NZ	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET	P	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET	PE	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET	PO	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RET	Z	5/11	1	2.5/5.5	1.4/3.1	1.3/2.8	
RETI		14	2	7	4	3.5	
RETN		14	2	7	4	3.5	
RLCA		4	1	2	1.1	1	
RLA		4	1	2	1.1	1	
RRCA		4	1	2	1.1	1	
RRA		4	1	2	1.1	1	
RLC	r	8	2	4	2.3	2	
RLC	(HL)	15	2	7.5	4.3	3.8	
RLC	(xy+d)	23	4	11.5	6.6	5.8	
RL	r	8	2	4	2.3	2	
RL	(HL)	15	2	7.5	4.3	3.8	
RL	(xy+d)	23	4	11.5	6.6	5.8	
RRC	r	8	2	4	2.3	2	
RRC	(HL)	15	2	7.5	4.3	3.8	
RRC	(xy+d)	23	4	11.5	6.6	5.8	
RR	r	8	2	4	2.3	2	
RR	(HL)	15	2	7.5	4.3	3.8	
RR	(xy+d)	23	4	11.5	6.6	5.8	
RLD		18	2	9	5.1	4.5	
RRD		18	2	9	5.1	4.5	
RST	a	11	1	5.5	3.1	2.8	a=(0,8,16,24,32, 40,48,56)
SBC	A,n	7	2	3.5	2	1.8	
SBC	A,r	4	1	2	1.1	1	
SBC	A,(HL)	7	1	3.5	2	1.8	
SBC	A,(xy+d)	19	3	9.5	5.4	4.8	
SBC	HL,rp	15	2	7.5	4.3	3.8	rp=(BC,DE,HL,SP)

		t	p	2 MHz	3.5 MHz	4 MHz	
SCF		4	1	2	1.1	1	
SET	b,r	8	2	4	2.3	2	
SET	b,(HL)	15	2	7.5	4.3	3.8	
SET	b,(xy+d)	23	4	11.5	6.6	5.8	
SLA	r	8	2	4	2.3	2	
SLA	(HL)	15	2	7.5	4.3	3.8	
SLA	(xy+d)	23	4	11.5	6.6	5.8	
SRA	r	8	2	4	2.3	2	
SRA	(HL)	15	2	7.5	4.3	3.8	
SRA	(xy+d)	23	4	11.5	6.6	5.8	
SRL	r	8	2	4	2.3	2	
SRL	(HL)	15	2	7.5	4.3	3.8	
SRL	(xy+d)	23	4	11.5	6.6	5.8	
SUB	n	7	2	3.5	2	1.8	
SUB	r	4	1	2	1.1	1	
SUB	(HL)	7	1	3.5	2	1.8	
SUB	(xy+d)	19	3	9.5	5.4	4.8	
XOR	n	7	2	3.5	2	1.8	
XOR	r	4	1	2	1.1	1	
XOR	(HL)	7	1	3.5	2	1.8	
XOR	(xy+d)	19	3	9.5	5.4	4.8	

DODATEK II.

TABELE SZESNASTKOWYCH KODÓW ROZKAZÓW MIKROPROCESORA Z80

Szesnastkowy kod określonego rozkazu jest wyznaczany przez sumowanie 2-cyfrowej liczby szesnastkowej (od 00 do F0) określającej wiersz w którym rozkaz jest zapisany z 1-cyfrową liczbą szesnastkową (od 0 do F) określającą kolumnę.
Przykładowo, rozkaz CALL nn, zapisany w wierszu C0 i kolumnie D tablicy D.II.1 identyfikowany jest szesnastkowym kodem C0 + D = CD.

Oznaczenia przyjęte w tablicach są analogiczne jak w dodatku I, oprócz różnic, które każdorazowo zaznaczono oraz innego zapisu argumentów rozkazów RST. Argumentami są liczby z zakresu od 0 do 7, wyznaczające odpowiednio adresy 0, 8, 16,... 56.

W tablicach zamieszczono także kody rozkazów nie opublikowanych w oficjalnych materiałach firmy Zilog oraz przypadki interpretacji przez mikroprocesor różnych kodów jako ten sam rozkaz (przykładowo liczba szesnastkowa 22 i dwie liczby szesnastkowe ED, 63 są interpretowane jako kod rozkazu LD (nn),HL). Rozkazom tym nadano symboliczne kody, ale nie są one akceptowane przez standardowe asemblery i disasemblery. W programach można je umieścić w postaci kodów liczbowych (w tablicach D.II.2. do D.II.5. zostały zaznaczone symbolem *).

Pola z wpisanym oznaczeniem tablicy oznaczają kody wielobajtowe (głównie 2-bajtowe), opisane w odpowiedniej tablicy.

Tablica D.II.1. Rozkazy o kodach 1-bajtowych

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	NOP	LD BC,nn	LD (BC),A	INC BC	INC B	DEC B	LD B,n	RLCA	EX AF,AF'	ADD HL,BC	LD A,(BC)	DEC BC	INC C	DEC C	LD C,n	FRCA
10	DJNZ n	LD DE,nn	LD (DE),A	INC DE	INC D	DEC D	LD D,n	RLA	JR n	ADD HL,DE	LD A,(DE)	DEC DE	INC E	DEC E	LD E,n	FRA
20	JR NZ,n	LD HL,nn	LD (nn),HL	INC HL	INC H	DEC H	LD H,n	DAA	JR Z,n	ADD HL,HL	LD HL,(nn)	DEC HL	INC L	DEC L	LD L,n	CPL
30	JR NC,n	LD SP,nn	LD (nn),A	INC SP	INC (HL)	DEC (HL)	LD (HL),n	SCF	JR C,n	ADD HL,SP	LD A,(nn)	DEC SP	INC A	DEC A	LD A,n	CCF
40	LD B,B	LD B,C	LD B,D	LD B,E	LD B,H	LD B,L	LD B,(HL)	LD B,A	LD C,B	LD C,C	LD C,D	LD C,E	LD C,H	LD C,L	LD C,(HL)	LD C,A
50	LD D,B	LD D,C	LD D,D	LD D,E	LD D,H	LD D,L	LD D,(HL)	LD D,A	LD E,B	LD E,C	LD E,D	LD E,E	LD E,H	LD E,L	LD E,(HL)	LD E,A
60	LD H,B	LD H,C	LD H,D	LD H,E	LD H,H	LD H,L	LD H,(HL)	LD H,A	LD L,B	LD L,C	LD L,D	LD L,E	LD L,H	LD L,L	LD L,(HL)	LD L,A
70	LD (HL),B	LD (HL),C	LD (HL),D	LD (HL),E	LD (HL),H	LD (HL),L	HALT	LD (HL),A	LD A,B	LD A,C	LD A,D	LD A,E	LD A,H	LD A,L	LD A,(HL)	LD A,A
80	ADD A,B	ADD A,C	ADD A,D	ADD A,E	ADD A,H	ADD A,L	ADD A,(HL)	ADD A,A	ADC A,B	ADC A,C	ADC A,D	ADC A,E	ADC A,H	ADC A,L	ADC A,(HL)	ADC A,A
90	SUB A,B	SUB A,C	SUB A,D	SUB A,E	SUB A,H	SUB A,L	SUB A,(HL)	SUB A,A	SBC A,B	SBC A,C	SBC A,D	SBC A,E	SBC A,H	SBC A,L	SBC A,(HL)	SBC A,A
A0	AND A,B	AND A,C	AND A,D	AND A,E	AND A,H	AND A,L	AND A,(HL)	AND A,A	XOR A,B	XOR A,C	XOR A,D	XOR A,E	XOR A,H	XOR A,L	XOR A,(HL)	XOR A,A
B0	OR A,B	OR A,C	OR A,D	OR A,E	OR A,H	OR A,L	OR A,(HL)	OR A,A	CP A,B	CP A,C	CP A,D	CP A,E	CP A,H	CP A,L	CP A,(HL)	CP A,A
C0	RET NZ	POP BC	JP NZ,nn	JP nn	CALL NZ,nn	PUSH BC	ADD A,n	RST 0	RET Z	RET	JP Z,nn	D.II.3.	CALL Z,nn	CALL nn	ADC A,n	RST 1
D0	RET NC	POP DE	JP NC,nn	OUT (n),A	CALL NC,nn	PUSH DE	SUB A,n	RST 2	RET C	EXX	JP C,nn	IN (n),A	CALL C,nn	D.II.4.	SBC A,n	RST 3
E0	RET PO	POP HL	JP PO,nn	EX (SP),HL	CALL PO,nn	PUSH HL	AND A,n	RST 4	RET PE	JP (HL)	JP PE,nn	EX DE,HL	CALL PE,nn	D.II.2.	XOR A,n	RST 5
F0	RET P	POP AF	JP P,nn	DI	CALL P,nn	PUSH AF	OR A,n	RST 6	RET M	LD SP,HL	JP M,nn	EI	CALL M,nn	D.II.5.	CP A,n	RST 7

Tablica D.II.2. Kody rozkazów poprzedzone bajtem ED

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00																
10																
20																
30																
40	IN B,(C)	OUT (C),B	SBC HL,BC	LD (nn),BC	NEG	RETN	IM 0	LD I,A	IN C,(C)	OUT (C),C	ADC HL,BC	LD BC,(nn)	* NEG	RETI		LD R,A
50	IN D,(C)	OUT (C),D	SBC HL,DE	LD (nn),DE	* NEG	* RETN	IM 1	LD A,I	IN E,(C)	OUT (C),E	ADC HL,DE	LD DE,(nn)	* NEG	* RETI	IM 2	LD A,R
60	IN H,(C)	OUT (C),H	SBC HL,HL	* LD (nn),HL	* NEG	* RETN		RRD	IN L,(C)	OUT (C),L	ADC HL,HL	* LD HL,(nn)	* NEG	* RETI		RLD
70	IN F,(C)		SBC HL,SP	LD (nn),SP	* NEG	* RETN			IN A,(C)	OUT (C),A	ADC HL,SP	LD SP,(nn)	* NEG	* RETI		
80																
90																
A0	LDI	CPI	INI	OUTI					LDD	CPD	IND	OUTD				
B0	LDIR	CPIR	INIR	OTIR					LDDR	CPDR	INDR	OTDR				
C0																
D0																
E0																
F0																

Kody odpowiadające pozycjom nie wypełnionym powyższej tablicy są interpretowane przez mikroprocesor jednakowo jako rozkaz NOP. Kody ED 63 i ED 6B nie są ujęte w firmowej liście rozkazów, asemblery generują odpowiednio 1-bajtowe kody 22 i 2A (wg tablicy D.II.1.).

Dla rozkazów NEG, RETN i RETI generowane są kody: ED 44, ED 45 i ED 4D. Pozostałe kody dla tych rozkazów należą do grupy nie opublikowanych.

Tablica D.II.3. Kody rozkazów poprzedzone bajtem CB

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	RLC B	RLC C	RLC D	RLC E	RLC H	RLC L	RLC (HL)	RLC A	RRC B	RRC C	RRC D	RRC E	RRC H	RRC L	RRC (HL)	RRC A
10	RL B	RL C	RL D	RL E	RL H	RL L	RL (HL)	RL A	RR B	RR C	RR D	RR E	RR H	RR L	RR (HL)	RR A
20	SLA B	SLA C	SLA D	SLA E	SLA H	SLA L	SLA (HL)	SLA A	SRA B	SRA C	SRA D	SRA E	SRA H	SRA L	SRA (HL)	SRA A
30	* SLA1 B	* SLA1 C	* SLA1 D	* SLA1 E	* SLA1 H	* SLA1 L	* SLA1 (HL)	* SLA1 A	SRL B	SRL C	SRL D	SRL E	SRL H	SRL L	SRL (HL)	SRL A
40	BIT 0,B	BIT 0,C	BIT 0,D	BIT 0,E	BIT 0,H	BIT 0,L	BIT 0,(HL)	BIT 0,A	BIT 1,B	BIT 1,C	BIT 1,D	BIT 1,E	BIT 1,H	BIT 1,L	BIT 1,(HL)	BIT 1,A
50	BIT 2,B	BIT 2,C	BIT 2,D	BIT 2,E	BIT 2,H	BIT 2,L	BIT 2,(HL)	BIT 2,A	BIT 3,B	BIT 3,C	BIT 3,D	BIT 3,E	BIT 3,H	BIT 3,L	BIT 3,(HL)	BIT 3,A
60	BIT 4,B	BIT 4,C	BIT 4,D	BIT 4,E	BIT 4,H	BIT 4,L	BIT 4,(HL)	BIT 4,A	BIT 5,B	BIT 5,C	BIT 5,D	BIT 5,E	BIT 5,H	BIT 5,L	BIT 5,(HL)	BIT 5,A
70	BIT 6,B	BIT 6,C	BIT 6,D	BIT 6,E	BIT 6,H	BIT 6,L	BIT 6,(HL)	BIT 6,A	BIT 7,B	BIT 7,C	BIT 7,D	BIT 7,E	BIT 7,H	BIT 7,L	BIT 7,(HL)	BIT 7,A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
80	RES 0,B	RES 0,C	RES 0,D	RES 0,E	RES 0,H	RES 0,L	RES 0,(HL)	RES 0,A	RES 1,B	RES 1,C	RES 1,D	RES 1,E	RES 1,H	RES 1,L	RES 1,(HL)	RES 1,A
90	RES 2,B	RES 2,C	RES 2,D	RES 2,E	RES 2,H	RES 2,L	RES 2,(HL)	RES 2,A	RES 3,B	RES 3,C	RES 3,D	RES 3,E	RES 3,H	RES 3,L	RES 3,(HL)	RES 3,A
A0	RES 4,B	RES 4,C	RES 4,D	RES 4,E	RES 4,H	RES 4,L	RES 4,(HL)	RES 4,A	RES 5,B	RES 5,C	RES 5,D	RES 5,E	RES 5,H	RES 5,L	RES 5,(HL)	RES 5,A
B0	RES 6,B	RES 6,C	RES 6,D	RES 6,E	RES 6,H	RES 6,L	RES 6,(HL)	RES 6,A	RES 7,B	RES 7,C	RES 7,D	RES 7,E	RES 7,H	RES 7,L	RES 7,(HL)	RES 7,A
C0	SET 0,B	SET 0,C	SET 0,D	SET 0,E	SET 0,H	SET 0,L	SET 0,(HL)	SET 0,A	SET 1,B	SET 1,C	SET 1,D	SET 1,E	SET 1,H	SET 1,L	SET 1,(HL)	SET 1,A
D0	SET 2,B	SET 2,C	SET 2,D	SET 2,E	SET 2,H	SET 2,L	SET 2,(HL)	SET 2,A	SET 3,B	SET 3,C	SET 3,D	SET 3,E	SET 3,H	SET 3,L	SET 3,(HL)	SET 3,A
E0	SET 4,B	SET 4,C	SET 4,D	SET 4,E	SET 4,H	SET 4,L	SET 4,(HL)	SET 4,A	SET 5,B	SET 5,C	SET 5,D	SET 5,E	SET 5,H	SET 5,L	SET 5,(HL)	SET 5,A
F0	SET 6,B	SET 6,C	SET 6,D	SET 6,E	SET 6,H	SET 6,L	SET 6,(HL)	SET 6,A	SET 7,B	SET 7,C	SET 7,D	SET 7,E	SET 7,H	SET 7,L	SET 7,(HL)	SET 7,A

SLA1 - nie opublikowany rozkaz podobny w działaniu do SLA, różniący się od niego wprowadzeniem "1" zamiast "0" na zerową pozycję binarną argumentu.

Tablica D.II.4. Kody rozkazów poprzedzone bajtem DD

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	* NOP	* LD BC,nn	* LD (BC),A	* INC BC	* INC B	* DEC B	* LD B,n	* RLCA	* EX AF,AF'	ADD X,BC	* LD A,(BC)	* DEC BC	* INC C	* DEC C	* LD C,n	* RRCA
10	* DJNZ n	* LD DE,nn	* LD (DE),A	* INC DE	* INC D	* DEC D	* LD D,n	* RLA	* JR n	ADD X,DE	* LD A,(DE)	* DEC DE	* INC E	* DEC E	* LD E,n	* RRA
20	* JR NZ,n	LD X,nn	LD (nn),X	INC X	* INC HX	* DEC HX	* LD HX,n	* DAA	* JR Z,n	ADD X,X	LD X,(nn)	DEC X	* INC LX	* DEC LX	* LD LX,n	* CPL
30	* JR NC,n	* LD SP,nn	* LD (nn),A	* INC SP	INC (X+d)	DEC (X+d)	LD (X+d),n	* SCF	* JR C,n	ADD X,SP	* LD A,(nn)	* DEC SP	* INC A	* DEC A	* LD A,n	* CCF
40	* LD B,B	* LD B,C	* LD B,D	* LD B,E	* LD B,HX	* LD B,LX	LD B,(X+d)	* LD B,A	* LD C,B	* LD C,C	* LD C,D	* LD C,E	* LD C,HX	* LD C,LX	LD C,(X+d)	* LD C,A
50	* LD D,B	* LD D,C	* LD D,D	* LD D,E	* LD D,HX	* LD D,LX	LD D,(X+d)	* LD D,A	* LD E,B	* LD E,C	* LD E,D	* LD E,E	* LD E,HX	* LD E,LX	LD E,(X+d)	* LD E,A
60	* LD HX,B	* LD HX,C	* LD HX,D	* LD HX,E	* LD HX,HX	* LD HX,LX	LD H,(X+d)	* LD HX,A	* LD LX,B	* LD LX,C	* LD LX,D	* LD LX,E	* LD LX,HX	* LD LX,LX	LD L,(X+d)	* LD LX,A
70	LD (X+d),B	LD (X+d),C	LD (X+d),D	LD (X+d),E	LD (X+d),H	LD (X+d),L	* HALT	LD (X+d),A	* LD A,B	* LD A,C	* LD A,D	* LD A,E	* LD A,HX	* LD A,LX	LD A,(X+d)	* LD A,A
80	* ADD A,B	* ADD A,C	* ADD A,D	* ADD A,E	* ADD A,HX	* ADD A,LX	ADD A,(X+d)	* ADD A,A	* ADC A,B	* ADC A,C	* ADC A,D	* ADC A,E	* ADC A,HX	* ADC A,LX	ADC A,(X+d)	* ADC A,A
90	* SUB A,B	* SUB A,C	* SUB A,D	* SUB A,E	* SUB A,HX	* SUB A,LX	SUB A,(X+d)	* SUB A,A	* SBC A,B	* SBC A,C	* SBC A,D	* SBC A,E	* SBC A,HX	* SBC A,LX	SBC A,(X+d)	* SBC A,A
A0	* AND A,B	* AND A,C	* AND A,D	* AND A,E	* AND A,HX	* AND A,LX	AND A,(X+d)	* AND A,A	* XOR A,B	* XOR A,C	* XOR A,D	* XOR A,E	* XOR A,HX	* XOR A,LX	XOR A,(X+d)	* XOR A,A
B0	* OR A,B	* OR A,C	* OR A,D	* OR A,E	* OR A,HX	* OR A,LX	OR A,(X+d)	* OR A,A	* CP A,B	* CP A,C	* CP A,D	* CP A,E	* CP A,HX	* CP A,LX	CP A,(X+d)	* CP A,A
C0	* RET NZ	* POP BC	* JP NZ,nn	* JP nn	* CALL NZ,nn	* PUSH BC	* ADD A,n	* RST 0	* RET Z	* RET	* JP Z,nn	D.II.6.	* CALL Z,nn	* CALL nn	* ADC A,n	* RST 1
D0	* RET NC	* POP DE	* JP NC,nn	* OUT (n),A	* CALL NC,nn	* PUSH DE	* SUB A,n	* RST 2	* RET C	* EXX	* JP C,nn	* IN (n),A	* CALL C,nn	D.II.4.	* SBC A,n	* RST 3
E0	* RET PO	POP X	* JP PO,nn	EX (SP),X	* CALL PO,nn	PUSH X	* AND A,n	* RST 4	* RET PE	JP (X)	* JP PE,nn	* EX DE,HL	* CALL PE,nn	D.II.2.	* XOR A,n	* RST 5
F0	* RET P	* POP AF	* JP P,nn	* DI	* CALL P,nn	* PUSH AF	* OR A,n	* RST 6	* RET M	LD SP,X	* JP M,nn	* EI	* CALL M,nn	D.II.5.	* CP A,n	* RST 7

Wszystkie rozkazy operujące na zawartości połowy rejestru IX oraz będące powtórzeniami z tablicy D.II.1 są nie opublikowane. W ciągach kodów DD DD, DD ED i DD FD pierwszy bajt jest pomijany (traktowany jako rozkaz NOP), drugi wchodzi w skład rozkazu odpowiednio wg tablicy D.II.4. (DD), D.II.2 (ED) i D.Ii.5. (FD).

X - 16-bitowy rejestr IX

LX - 8 mniej znaczących pozycji binarnych rejestru IX

HX - 8 bardziej znaczących pozycji binarnych rejestru IX

Tablica D.II.5. Kody rozkazów poprzedzone bajtem FD

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	* NOP	* LD BC,nn	* LD (BC),A	* INC BC	* INC B	* DEC B	* LD B,n	* RLCA	* EX AF,AF'	ADD Y,BC	* LD A,(BC)	* DEC BC	* INC C	* DEC C	* LD C,n	* RRCA
10	* DJNZ n	* LD DE,nn	* LD (DE),A	* INC DE	* INC D	* DEC D	* LD D,n	* RLA	* JR n	ADD Y,DE	* LD A,(DE)	* DEC DE	* INC E	* DEC E	* LD E,n	* RRA
20	* JR NZ,n	LD Y,nn	LD (nn),Y	INC Y	* INC HY	* DEC HY	* LD HY,n	* DAA	* JR Z,n	ADD Y,Y	LD Y,(nn)	DEC Y	* INC LY	* DEC LY	* LD LY,n	* CPL
30	* JR NC,n	* LD SP,nn	* LD (nn),A	* INC SP	INC (Y+d)	DEC (Y+d)	LD (Y+d),n	* SCF	* JR C,n	ADD Y,SP	* LD A,(nn)	* DEC SP	* INC A	* DEC A	* LD A,n	* CCF
40	* LD B,B	* LD B,C	* LD B,D	* LD B,E	* LD B,HY	* LD B,LY	LD B,(Y+d)	* LD B,A	* LD C,B	* LD C,C	* LD C,D	* LD C,E	* LD C,HY	* LD C,LY	LD C,(Y+d)	* LD C,A
50	* LD D,B	* LD D,C	* LD D,D	* LD D,E	* LD D,HY	* LD D,LY	LD D,(Y+d)	* LD D,A	* LD E,B	* LD E,C	* LD E,D	* LD E,E	* LD E,HY	* LD E,LY	LD E,(Y+d)	* LD E,A
60	* LD HY,B	* LD HY,C	* LD HY,D	* LD HY,E	* LD HY,HY	* LD HY,LY	LD H,(Y+d)	* LD HY,A	* LD LY,B	* LD LY,C	LD LY,D	LD LY,E	LD LY,HY	LD LY,LY	LD L,(Y+d)	LD LY,A
70	LD (Y+d),B	LD (Y+d),C	LD (Y+d),D	LD (Y+d),E	LD (Y+d),H	LD (Y+d),L	* HALT	LD (Y+d),A	* LD A,B	* LD A,C	* LD A,D	* LD A,E	* LD A,HY	* LD A,LY	LD A,(Y+d)	* LD A,A
80	* ADD A,B	* ADD A,C	* ADD A,D	* ADD A,E	* ADD A,HY	* ADD A,LY	ADD A,(Y+d)	* ADD A,A	* ADC A,B	* ADC A,C	* ADC A,D	* ADC A,E	* ADC A,HY	* ADC A,LY	ADC A,(Y+d)	* ADC A,A
90	* SUB A,B	* SUB A,C	* SUB A,D	* SUB A,E	* SUB A,HY	* SUB A,LY	SUB A,(Y+d)	* SUB A,A	* SBC A,B	* SBC A,C	* SBC A,D	* SBC A,E	* SBC A,HY	* SBC A,LY	SBC A,(Y+d)	* SBC A,A
A0	* AND A,B	* AND A,C	* AND A,D	* AND A,E	* AND A,HY	* AND A,LY	AND A,(Y+d)	* AND A,A	* XOR A,B	* XOR A,C	* XOR A,D	* XOR A,E	* XOR A,HY	* XOR A,LY	XOR A,(Y+d)	* XOR A,A
B0	* OR A,B	* OR A,C	* OR A,D	* OR A,E	* OR A,HY	* OR A,LY	OR A,(Y+d)	* OR A,A	* CP A,B	* CP A,C	* CP A,D	* CP A,E	* CP A,HY	* CP A,LY	CP A,(Y+d)	* CP A,A
C0	* RET NZ	* POP BC	* JP NZ,nn	* JP nn	* CALL NZ,nn	* PUSH BC	* ADD A,n	* RST 0	* RET Z	* RET	* JP Z,nn	D.II.7.	* CALL Z,nn	* CALL nn	* ADC A,n	* RST 1
D0	* RET NC	* POP DE	* JP NC,nn	* OUT (n),A	* CALL NC,nn	* PUSH DE	* SUB A,n	* RST 2	* RET C	* EXX	* JP C,nn	* IN (n),A	* CALL C,nn	D.II.4.	* SBC A,n	* RST 3
E0	* RET PO	POP Y	* JP PO,nn	EX (SP),Y	* CALL PO,nn	PUSH Y	* AND A,n	* RST 4	* RET PE	JP (Y)	* JP PE,nn	EX DE,HL	* CALL PE,nn	D.II.2.	* XOR A,n	* RST 5
F0	* RET P	* POP AF	* JP P,nn	* DI	* CALL P,nn	* PUSH AF	* OR A,n	* RST 6	* RET M	* LD SP,Y	* JP M,nn	* EI	* CALL M,nn	D.II.5.	* CP A,n	* RST 7

Wszystkie rozkazy operujące na zawartości połowy rejestru IY oraz będące powtórzeniami z tablicy D.II.1 są nie opublikowane. W ciągach kodów FD DD, FD ED i FD FD pierwszy bajt jest pomijany (traktowany jako rozkaz NOP), drugi wchodzi w skład rozkazu odpowiednio wg tablicy D.II.4. (DD), D.II.2 (ED) i D.II.5. (FD).

X - 16-bitowy rejestr IY

LY - 8 mniej znaczących pozycji binarnych rejestru IY

HY - 8 bardziej znaczących pozycji binarnych rejestru IY

Tablica D.II.6. Kody rozkazów poprzedzone bajtami DD i CB

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	RLC:B (X+d)	RLC:C (X+d)	RLC:D (X+d)	RLC:E (X+d)	RLC:H (X+d)	RLC:L (X+d)	RLC (X+d)	RLC:A (X+d)	RRC:B (X+d)	RRC:C (X+d)	RRC:D (X+d)	RRC:E (X+d)	RRC:H (X+d)	RRC:L (X+d)	RRC (X+d)	RRC:A (X+d)
10	RL:B (X+d)	RL:C (X+d)	RL:D (X+d)	RL:E (X+d)	RL:H (X+d)	RL:L (X+d)	RL (X+d)	RL:A (X+d)	RR:B (X+d)	RR:C (X+d)	RR:D (X+d)	RR:E (X+d)	RR:H (X+d)	RR:L (X+d)	RR (X+d)	RR:A (X+d)
20	SLA:B (X+d)	SLA:C (X+d)	SLA:D (X+d)	SLA:E (X+d)	SLA:H (X+d)	SLA:L (X+d)	SLA (X+d)	SLA:A (X+d)	SRA:B (X+d)	SRA:C (X+d)	SRA:D (X+d)	SRA:E (X+d)	SRA:H (X+d)	SRA:L (X+d)	SRA (X+d)	SRA:A (X+d)
30	SLL:B (X+d)	SLL:C (X+d)	SLL:D (X+d)	SLL:E (X+d)	SLL:H (X+d)	SLL:L (X+d)	SLL (X+d)	SLL:A (X+d)	SRL:B (X+d)	SRL:C (X+d)	SRL:D (X+d)	SRL:E (X+d)	SRL:H (X+d)	SRL:L (X+d)	SRL (X+d)	SRL:A (X+d)
40	BIT:B 0,(X+d)	BIT:C 0,(X+d)	BIT:D 0,(X+d)	BIT:E 0,(X+d)	BIT:H 0,(X+d)	BIT:L 0,(X+d)	BIT 0,(X+d)	BIT:A 0,(X+d)	BIT:B 1,(X+d)	BIT:C 1,(X+d)	BIT:D 1,(X+d)	BIT:E 1,(X+d)	BIT:H 1,(X+d)	BIT:L 1,(X+d)	BIT 1,(X+d)	BIT:A 1,(X+d)
50	BIT:B 2,(X+d)	BIT:C 2,(X+d)	BIT:D 2,(X+d)	BIT:E 2,(X+d)	BIT:H 2,(X+d)	BIT:L 2,(X+d)	BIT 2,(X+d)	BIT:A 2,(X+d)	BIT:B 3,(X+d)	BIT:C 3,(X+d)	BIT:D 3,(X+d)	BIT:E 3,(X+d)	BIT:H 3,(X+d)	BIT:L 3,(X+d)	BIT 3,(X+d)	BIT:A 3,(X+d)
60	BIT:B 4,(X+d)	BIT:C 4,(X+d)	BIT:D 4,(X+d)	BIT:E 4,(X+d)	BIT:H 4,(X+d)	BIT:L 4,(X+d)	BIT 4,(X+d)	BIT:A 4,(X+d)	BIT:B 5,(X+d)	BIT:C 5,(X+d)	BIT:D 5,(X+d)	BIT:E 5,(X+d)	BIT:H 5,(X+d)	BIT:L 5,(X+d)	BIT 5,(X+d)	BIT:A 5,(X+d)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
70	BIT:B 6,(X+d)	BIT:C 6,(X+d)	BIT:D 6,(X+d)	BIT:E 6,(X+d)	BIT:H 6,(X+d)	BIT:L 6,(X+d)	BIT 6,(X+d)	BIT:A 6,(X+d)	BIT:B 7,(X+d)	BIT:C 7,(X+d)	BIT:D 7,(X+d)	BIT:E 7,(X+d)	BIT:H 7,(X+d)	BIT:L 7,(X+d)	BIT 7,(X+d)	BIT:A 7,(X+d)
80	RES:B 0,(X+d)	RES:C 0,(X+d)	RES:D 0,(X+d)	RES:E 0,(X+d)	RES:H 0,(X+d)	RES:L 0,(X+d)	RES 0,(X+d)	RES:A 0,(X+d)	RES:B 1,(X+d)	RES:C 1,(X+d)	RES:D 1,(X+d)	RES:E 1,(X+d)	RES:H 1,(X+d)	RES:L 1,(X+d)	RES 1,(X+d)	RES:A 1,(X+d)
90	RES:B 2,(X+d)	RES:C 2,(X+d)	RES:D 2,(X+d)	RES:E 2,(X+d)	RES:H 2,(X+d)	RES:L 2,(X+d)	RES 2,(X+d)	RES:A 2,(X+d)	RES:B 3,(X+d)	RES:C 3,(X+d)	RES:D 3,(X+d)	RES:E 3,(X+d)	RES:H 3,(X+d)	RES:L 3,(X+d)	RES 3,(X+d)	RES:A 3,(X+d)
A0	RES:B 4,(X+d)	RES:C 4,(X+d)	RES:D 4,(X+d)	RES:E 4,(X+d)	RES:H 4,(X+d)	RES:L 4,(X+d)	RES 4,(X+d)	RES:A 4,(X+d)	RES:B 5,(X+d)	RES:C 5,(X+d)	RES:D 5,(X+d)	RES:E 5,(X+d)	RES:H 5,(X+d)	RES:L 5,(X+d)	RES 5,(X+d)	RES:A 5,(X+d)
B0	RES:B 6,(X+d)	RES:C 6,(X+d)	RES:D 6,(X+d)	RES:E 6,(X+d)	RES:H 6,(X+d)	RES:L 6,(X+d)	RES 6,(X+d)	RES:A 6,(X+d)	RES:B 7,(X+d)	RES:C 7,(X+d)	RES:D 7,(X+d)	RES:E 7,(X+d)	RES:H 7,(X+d)	RES:L 7,(X+d)	RES 7,(X+d)	RES:A 7,(X+d)
C0	SET:B 0,(X+d)	SET:C 0,(X+d)	SET:D 0,(X+d)	SET:E 0,(X+d)	SET:H 0,(X+d)	SET:L 0,(X+d)	SET 0,(X+d)	SET:A 0,(X+d)	SET:B 1,(X+d)	SET:C 1,(X+d)	SET:D 1,(X+d)	SET:E 1,(X+d)	SET:H 1,(X+d)	SET:L 1,(X+d)	SET 1,(X+d)	SET:A 1,(X+d)
D0	SET:B 2,(X+d)	SET:C 2,(X+d)	SET:D 2,(X+d)	SET:E 2,(X+d)	SET:H 2,(X+d)	SET:L 2,(X+d)	SET 2,(X+d)	SET:A 2,(X+d)	SET:B 3,(X+d)	SET:C 3,(X+d)	SET:D 3,(X+d)	SET:E 3,(X+d)	SET:H 3,(X+d)	SET:L 3,(X+d)	SET 3,(X+d)	SET:A 3,(X+d)
E0	SET:B 4,(X+d)	SET:C 4,(X+d)	SET:D 4,(X+d)	SET:E 4,(X+d)	SET:H 4,(X+d)	SET:L 4,(X+d)	SET 4,(X+d)	SET:A 4,(X+d)	SET:B 5,(X+d)	SET:C 5,(X+d)	SET:D 5,(X+d)	SET:E 5,(X+d)	SET:H 5,(X+d)	SET:L 5,(X+d)	SET 5,(X+d)	SET:A 5,(X+d)
F0	SET:B 6,(X+d)	SET:C 6,(X+d)	SET:D 6,(X+d)	SET:E 6,(X+d)	SET:H 6,(X+d)	SET:L 6,(X+d)	SET 6,(X+d)	SET:A 6,(X+d)	SET:B 7,(X+d)	SET:C 7,(X+d)	SET:D 7,(X+d)	SET:E 7,(X+d)	SET:H 7,(X+d)	SET:L 7,(X+d)	SET 7,(X+d)	SET:A 7,(X+d)

Wszystkie rozkazy zaznaczone rejestrem poprzedzonym znakiem : są nie opublikowane. Wprowadzona notacja oznacza, że rozkaz wykonuje się na argumencie określonym w dolnej pozycji pola tablicy, a dodatkowo uzyskany wynik jest przesyłany do tak zaznaczonego rejestru. Jedynie rozkazy z kolumny 6 i E należą do grupy opublikowanych. X - rejestr IX

Tablica D.II.7. Kody rozkazów poprzedzone bajtami FD i CB

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	RLC:B (Y+d)	RLC:C (Y+d)	RLC:D (Y+d)	RLC:E (Y+d)	RLC:H (Y+d)	RLC:L (Y+d)	RLC (Y+d)	RLC:A (Y+d)	RRC:B (Y+d)	RRC:C (Y+d)	RRC:D (Y+d)	RRC:E (Y+d)	RRC:H (Y+d)	RRC:L (Y+d)	RRC (Y+d)	RRC:A (Y+d)
10	RL:B (Y+d)	RL:C (Y+d)	RL:D (Y+d)	RL:E (Y+d)	RL:H (Y+d)	RL:L (Y+d)	RL (Y+d)	RL:A (Y+d)	RR:B (Y+d)	RR:C (Y+d)	RR:D (Y+d)	RR:E (Y+d)	RR:H (Y+d)	RR:L (Y+d)	RR (Y+d)	RR:A (Y+d)
20	SLA:B (Y+d)	SLA:C (Y+d)	SLA:D (Y+d)	SLA:E (Y+d)	SLA:H (Y+d)	SLA:L (Y+d)	SLA (Y+d)	SLA:A (Y+d)	SRA:B (Y+d)	SRA:C (Y+d)	SRA:D (Y+d)	SRA:E (Y+d)	SRA:H (Y+d)	SRA:L (Y+d)	SRA (Y+d)	SRA:A (Y+d)
30	SLL:B (Y+d)	SLL:C (Y+d)	SLL:D (Y+d)	SLL:E (Y+d)	SLL:H (Y+d)	SLL:L (Y+d)	SLL (Y+d)	SLL:A (Y+d)	SRL:B (Y+d)	SRL:C (Y+d)	SRL:D (Y+d)	SRL:E (Y+d)	SRL:H (Y+d)	SRL:L (Y+d)	SRL (Y+d)	SRL:A (Y+d)
40	BIT:B 0,(Y+d)	BIT:C 0,(Y+d)	BIT:D 0,(Y+d)	BIT:E 0,(Y+d)	BIT:H 0,(Y+d)	BIT:L 0,(Y+d)	BIT 0,(Y+d)	BIT:A 0,(Y+d)	BIT:B 1,(Y+d)	BIT:C 1,(Y+d)	BIT:D 1,(Y+d)	BIT:E 1,(Y+d)	BIT:H 1,(Y+d)	BIT:L 1,(Y+d)	BIT 1,(Y+d)	BIT:A 1,(Y+d)
50	BIT:B 2,(Y+d)	BIT:C 2,(Y+d)	BIT:D 2,(Y+d)	BIT:E 2,(Y+d)	BIT:H 2,(Y+d)	BIT:L 2,(Y+d)	BIT 2,(Y+d)	BIT:A 2,(Y+d)	BIT:B 3,(Y+d)	BIT:C 3,(Y+d)	BIT:D 3,(Y+d)	BIT:E 3,(Y+d)	BIT:H 3,(Y+d)	BIT:L 3,(Y+d)	BIT 3,(Y+d)	BIT:A 3,(Y+d)
60	BIT:B 4,(Y+d)	BIT:C 4,(Y+d)	BIT:D 4,(Y+d)	BIT:E 4,(Y+d)	BIT:H 4,(Y+d)	BIT:L 4,(Y+d)	BIT 4,(Y+d)	BIT:A 4,(Y+d)	BIT:B 5,(Y+d)	BIT:C 5,(Y+d)	BIT:D 5,(Y+d)	BIT:E 5,(Y+d)	BIT:H 5,(Y+d)	BIT:L 5,(Y+d)	BIT 5,(Y+d)	BIT:A 5,(Y+d)
70	BIT:B 6,(Y+d)	BIT:C 6,(Y+d)	BIT:D 6,(Y+d)	BIT:E 6,(Y+d)	BIT:H 6,(Y+d)	BIT:L 6,(Y+d)	BIT 6,(Y+d)	BIT:A 6,(Y+d)	BIT:B 7,(Y+d)	BIT:C 7,(Y+d)	BIT:D 7,(Y+d)	BIT:E 7,(Y+d)	BIT:H 7,(Y+d)	BIT:L 7,(Y+d)	BIT 7,(Y+d)	BIT:A 7,(Y+d)
B0	RES:B 0,(Y+d)	RES:C 0,(Y+d)	RES:D 0,(Y+d)	RES:E 0,(Y+d)	RES:H 0,(Y+d)	RES:L 0,(Y+d)	RES 0,(Y+d)	RES:A 0,(Y+d)	RES:B 1,(Y+d)	RES:C 1,(Y+d)	RES:D 1,(Y+d)	RES:E 1,(Y+d)	RES:H 1,(Y+d)	RES:L 1,(Y+d)	RES 1,(Y+d)	RES:A 1,(Y+d)
90	RES:B 2,(Y+d)	RES:C 2,(Y+d)	RES:D 2,(Y+d)	RES:E 2,(Y+d)	RES:H 2,(Y+d)	RES:L 2,(Y+d)	RES 2,(Y+d)	RES:A 2,(Y+d)	RES:B 3,(Y+d)	RES:C 3,(Y+d)	RES:D 3,(Y+d)	RES:E 3,(Y+d)	RES:H 3,(Y+d)	RES:L 3,(Y+d)	RES 3,(Y+d)	RES:A 3,(Y+d)
A0	RES:B 4,(Y+d)	RES:C 4,(Y+d)	RES:D 4,(Y+d)	RES:E 4,(Y+d)	RES:H 4,(Y+d)	RES:L 4,(Y+d)	RES 4,(Y+d)	RES:A 4,(Y+d)	RES:B 5,(Y+d)	RES:C 5,(Y+d)	RES:D 5,(Y+d)	RES:E 5,(Y+d)	RES:H 5,(Y+d)	RES:L 5,(Y+d)	RES 5,(Y+d)	RES:A 5,(Y+d)
B0	RES:B 6,(Y+d)	RES:C 6,(Y+d)	RES:D 6,(Y+d)	RES:E 6,(Y+d)	RES:H 6,(Y+d)	RES:L 6,(Y+d)	RES 6,(Y+d)	RES:A 6,(Y+d)	RES:B 7,(Y+d)	RES:C 7,(Y+d)	RES:D 7,(Y+d)	RES:E 7,(Y+d)	RES:H 7,(Y+d)	RES:L 7,(Y+d)	RES 7,(Y+d)	RES:A 7,(Y+d)
C0	SET:B 0,(Y+d)	SET:C 0,(Y+d)	SET:D 0,(Y+d)	SET:E 0,(Y+d)	SET:H 0,(Y+d)	SET:L 0,(Y+d)	SET 0,(Y+d)	SET:A 0,(Y+d)	SET:B 1,(Y+d)	SET:C 1,(Y+d)	SET:D 1,(Y+d)	SET:E 1,(Y+d)	SET:H 1,(Y+d)	SET:L 1,(Y+d)	SET 1,(Y+d)	SET:A 1,(Y+d)
D0	SET:B 2,(Y+d)	SET:C 2,(Y+d)	SET:D 2,(Y+d)	SET:E 2,(Y+d)	SET:H 2,(Y+d)	SET:L 2,(Y+d)	SET 2,(Y+d)	SET:A 2,(Y+d)	SET:B 3,(Y+d)	SET:C 3,(Y+d)	SET:D 3,(Y+d)	SET:E 3,(Y+d)	SET:H 3,(Y+d)	SET:L 3,(Y+d)	SET 3,(Y+d)	SET:A 3,(Y+d)
E0	SET:B 4,(Y+d)	SET:C 4,(Y+d)	SET:D 4,(Y+d)	SET:E 4,(Y+d)	SET:H 4,(Y+d)	SET:L 4,(Y+d)	SET 4,(Y+d)	SET:A 4,(Y+d)	SET:B 5,(Y+d)	SET:C 5,(Y+d)	SET:D 5,(Y+d)	SET:E 5,(Y+d)	SET:H 5,(Y+d)	SET:L 5,(Y+d)	SET 5,(Y+d)	SET:A 5,(Y+d)
F0	SET:B 6,(Y+d)	SET:C 6,(Y+d)	SET:D 6,(Y+d)	SET:E 6,(Y+d)	SET:H 6,(Y+d)	SET:L 6,(Y+d)	SET 6,(Y+d)	SET:A 6,(Y+d)	SET:B 7,(Y+d)	SET:C 7,(Y+d)	SET:D 7,(Y+d)	SET:E 7,(Y+d)	SET:H 7,(Y+d)	SET:L 7,(Y+d)	SET 7,(Y+d)	SET:A 7,(Y+d)

Wszystkie rozkazy zaznaczone rejestrem poprzedzonym znakiem : są nie opublikowane. Wprowadzona notacja oznacza, że rozkaz wykonuje się na argumencie określonym w dolnej pozycji pola tablicy, a dodatkowo uzyskany wynik jest przesyłany do tak zaznaczonego rejestru. Jedynie rozkazy z kolumny 6 i E należą do grupy opublikowanych. Y - rejestr IY

DODATEK III.

TABELE KONWERSJI LICZB W RÓŻNYCH SYSTEMACH LICZENIA

Tablica D.III.1. Interpretacja zawartości 8-bitowej komórki w różnych systemach liczenia i znakowo

dziesiętnie bez znaku	szesnastkowo	dziesiętnie ze znakiem	binarnie	znakowo w kodzie ASCII
0	00	0	00000000	NUL
1	01	1	00000001	SOH
2	02	2	00000010	STX
3	03	3	00000011	ETX
4	04	4	00000100	EOT
5	05	5	00000101	ENQ
6	06	6	00000110	ACK
7	07	7	00000111	BEL
8	08	8	00001000	BS
9	09	9	00001001	HT
10	0A	10	00001010	LF
11	0B	11	00001011	VT
12	0C	12	00001100	FF
13	0D	13	00001101	CR
14	0E	14	00001110	SO
15	0F	15	00001111	SI
16	10	16	00010000	DLE
17	11	17	00010001	DC1
18	12	18	00010010	DC2
19	13	19	00010011	DC3
20	14	20	00010100	DC4
21	15	21	00010101	NAK
22	16	22	00010110	SYN
23	17	23	00010111	ETB
24	18	24	00011000	CAN
25	19	25	00011001	EM
26	1A	26	00011010	SUB
27	1B	27	00011011	ESC
28	1C	28	00011100	FS
29	1D	29	00011101	GS
30	1E	30	00011110	RS
31	1F	31	00011111	US
32	20	32	00100000	spacja
33	21	33	00100001	!
34	22	34	00100010	"
35	23	35	00100011	#
36	24	36	00100100	$
37	25	37	00100101	%
38	26	38	00100110	&
39	27	39	00100111	'
40	28	40	00101000	(
41	29	41	00101001	)
42	2A	42	00101010	*
43	2B	43	00101011	+
44	2C	44	00101100	,
45	2D	45	00101101	-
46	2E	46	00101110	.
47	2F	47	00101111	/
48	30	48	00110000	0
49	31	49	00110001	1
50	32	50	00110010	2
51	33	51	00110011	3
52	34	52	00110100	4
53	35	53	00110101	5
54	36	54	00110110	6
55	37	55	00110111	7
56	38	56	00111000	8
57	39	57	00111001	9
58	3A	58	00111010	:
59	3B	59	00111011	;
60	3C	60	00111100	<
61	3D	61	00111101	=
62	3E	62	00111110	>
63	3F	63	00111111	?
64	40	64	01000000	@
65	41	65	01000001	A
66	42	66	01000010	B
67	43	67	01000011	C
68	44	68	01000100	D
69	45	69	01000101	E
70	46	70	01000110	F
71	47	71	01000111	G
72	48	72	01001000	H
73	49	73	01001001	I
74	4A	74	01001010	J
75	4B	75	01001011	K
76	4C	76	01001100	L
77	4D	77	01001101	M
78	4E	78	01001110	N
79	4F	79	01001111	O
80	50	80	01010000	P
81	51	81	01010001	Q
82	52	82	01010010	R
83	53	83	01010011	S
84	54	84	01010100	T
85	55	85	01010101	U
86	56	86	01010110	V
87	57	87	01010111	W
88	58	88	01011000	X
89	59	89	01011001	Y
90	5A	90	01011010	Z
91	5B	91	01011011	[
92	5C	92	01011100	\
93	5D	93	01011101	]
94	5E	94	01011110	^
95	5F	95	01011111	_
96	60	96	01100000	`
97	61	97	01100001	a
98	62	98	01100010	b
99	63	99	01100011	c
100	64	100	01100100	d
101	65	101	01100101	e
102	66	102	01100110	f
103	67	103	01100111	g
104	68	104	01101000	h
105	69	105	01101001	i
106	6A	106	01101010	j
107	6B	107	01101011	k
108	6C	108	01101100	l
109	6D	109	01101101	m
110	6E	110	01101110	n
111	6F	111	01101111	o
112	70	112	01110000	p
113	71	113	01110001	q
114	72	114	01110010	r
115	73	115	01110011	s
116	74	116	01110100	t
117	75	117	01110101	u
118	76	118	01110110	v
119	77	119	01110111	w
120	78	120	01111000	x
121	79	121	01111001	y
122	7A	122	01111010	z
123	7B	123	01111011	{
124	7C	124	01111100	|
125	7D	125	01111101	}
126	7E	126	01111110	~
127	7F	127	01111111	DEL
128	80	-128	10000000	
129	81	-127	10000001	
130	82	-126	10000010	
131	83	-125	10000011	
132	84	-124	10000100	
133	85	-123	10000101	
134	86	-122	10000110	
135	87	-121	10000111	
136	88	-120	10001000	

dziesiętnie bez znaku	szesnastkowo	dziesiętnie ze znakiem	binarnie	znakowo w kodzie ASCII
137	89	-119	10001001	
138	8A	-118	10001010	
139	8B	-117	10001011	
140	8C	-116	10001100	
141	8D	-115	10001101	
142	8E	-114	10001110	
143	8F	-113	10001111	
144	90	-112	10010000	
145	91	-111	10010001	
146	92	-110	10010010	
147	93	-109	10010011	
148	94	-108	10010100	
149	95	-107	10010101	
150	96	-106	10010110	
151	97	-105	10010111	
152	98	-104	10011000	
153	99	-103	10011001	
154	9A	-102	10011010	
155	9B	-101	10011011	
156	9C	-100	10011100	
157	9D	-99	10011101	
158	9E	-98	10011110	
159	9F	-97	10011111	
160	A0	-96	10100000	
161	A1	-95	10100001	
162	A2	-94	10100010	
163	A3	-93	10100011	
164	A4	-92	10100100	
165	A5	-91	10100101	
166	A6	-90	10100110	
167	A7	-89	10100111	
168	A8	-88	10101000	
169	A9	-87	10101001	
170	AA	-86	10101010	
171	AB	-85	10101011	
172	AC	-84	10101100	
173	AD	-83	10101101	
174	AE	-82	10101110	
175	AF	-81	10101111	
176	B0	-80	10110000	
177	B1	-79	10110001	
178	B2	-78	10110010	
179	B3	-77	10110011	
180	B4	-76	10110100	
181	B5	-75	10110101	
182	B6	-74	10110110	
183	B7	-73	10110111	
184	B8	-72	10111000	
185	B9	-71	10111001	
186	BA	-70	10111010	
187	BB	-69	10111011	
188	BC	-68	10111100	
189	BD	-67	10111101	
190	BE	-66	10111110	
191	BF	-65	10111111	
192	C0	-64	11000000	
193	C1	-63	11000001	
194	C2	-62	11000010	
195	C3	-61	11000011	
196	C4	-60	11000100	
197	C5	-59	11000101	
198	C6	-58	11000110	
199	C7	-57	11000111	
200	C8	-56	11001000	
201	C9	-55	11001001	
202	CA	-54	11001010	
203	CB	-53	11001011	
204	CC	-52	11001100	
205	CD	-51	11001101	
206	CE	-50	11001110	
207	CF	-49	11001111	
208	D0	-48	11010000	
209	D1	-47	11010001	
210	D2	-46	11010010	
211	D3	-45	11010011	
212	D4	-44	11010100	
213	D5	-43	11010101	
214	D6	-42	11010110	
215	D7	-41	11010111	
216	D8	-40	11011000	
217	D9	-39	11011001	
218	DA	-38	11011010	
219	DB	-37	11011011	
220	DC	-36	11011100	
221	DD	-35	11011101	
222	DE	-34	11011110	
223	DF	-33	11011111	
224	E0	-32	11100000	
225	E1	-31	11100001	
226	E2	-30	11100010	
227	E3	-29	11100011	
228	E4	-28	11100100	
229	E5	-27	11100101	
230	E6	-26	11100110	
231	E7	-25	11100111	
232	E8	-24	11101000	
233	E9	-23	11101001	
234	EA	-22	11101010	
235	EB	-21	11101011	
236	EC	-20	11101100	
237	ED	-19	11101101	
238	EE	-18	11101110	
239	EF	-17	11101111	
240	F0	-16	11110000	
241	F1	-15	11110001	
242	F2	-14	11110010	
243	F3	-13	11110011	
244	F4	-12	11110100	
245	F5	-11	11110101	
246	F6	-10	11110110	
247	F7	-9	11110111	
248	F8	-8	11111000	
249	F9	-7	11111001	
250	FA	-6	11111010	
251	FB	-5	11111011	
252	FC	-4	11111100	
253	FD	-3	11111101	
254	FE	-2	11111110	
255	FF	-1	11111111	

Tablica D.III.2. Konwersja liczb 1-bajtowych w systeeie dziesiętnym (bez znaku) i szesnastkowym

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
20	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
30	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
40	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
50	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
60	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
70	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
B0	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
90	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A0	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B0	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C0	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D0	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E0	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F0	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Liczby dziesiętne od 0 do 255 zamieszczone w teblicy są identyfikowane szesnastkowyei numereei wierszy (2-cyfrowe liczby od 00 do F0) i koluen (1-cyfrowe liczby od 0 do F).

Wyznaczenie wartości szesnastkowej wybranej liczby dziesiętnej polega na szesnastkowya dodaniu wartości określonego nuseru wiersza do koluemy. Przykładowo liczba dziesiętna 171, zapisana w wierszu A0 i koluenie B odpowiada liczbie AB (A0 + B) w systeeie szesnastkowym.

W wypadku liczby szesnastkowej, jej postać dziesiętną znajduje się w koluenie wyznaczonej przez eniej znaczącą cyfrę tej liczby i wierszu wyznaczonye przez odjęcie od tej liczby eniej znaczącej cyfry. Dla przykładu szesnastkowej liczbie 9A odpowiada liczba dziesiętna z koluemy A i wiersza 9A - A = 90, czyli 154.

LITERATURA

[1] Grabowski J., Koślacz S.: Podstawy i praktyka programowania mikroprocesorów. Warszawa, WNT 1980.

[2] Lipowski J., Małysiak H. i in.: Modułowe systemy mikrokomputerowe. Warszawa, WNT 1984.

[3] Misiurewicz P.: Układy mikroprocesorowe. Struktury i programowanie. Warszawa, WNT 1983.

[4] Sacha K., Rydzewski A.: Mikroprocesor w pytaniach i odpowiedziach. Warszawa, WNT 1985.

[5] Knuth D.: The art of computer programming. Fundamental algorithms. Vol. 1, Addison-Wesley, 1968.

[6] Knuth D.: The art of computer programming. Searching and sorting. Vol. 3, Addison-Wesley, 1973.

[7] Leventhal L., Saville W., : Z80 assembly language subroutines. Osborne/McGraw-Hill, 1983.

Tablica D.III.3. Konwersja liczb 2-bajtowych w systemie dziesiętnym (bez znaku) i szesnastkowym

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0 0	1 256	2 512	3 768	4 1024	5 1280	6 1536	7 1792	8 2048	9 2304	10 2560	11 2816	12 3072	13 3328	14 3584	15 3840
10	16 4096	17 4352	18 4608	19 4864	20 5120	21 5376	22 5632	23 5888	24 6144	25 6400	26 6656	27 6912	28 7168	29 7424	30 7680	31 7936
20	32 8192	33 8448	34 8704	35 8960	36 9216	37 9472	38 9728	39 9984	40 10240	41 10496	42 10752	43 11008	44 11264	45 11520	46 11776	47 12032
30	48 12288	49 12544	50 12800	51 13056	52 13312	53 13568	54 13824	55 14080	56 14336	57 14592	58 14848	59 15104	60 15360	61 15616	62 15872	63 16128
40	64 16384	65 16640	66 16896	67 17152	68 17408	69 17664	70 17920	71 18176	72 18432	73 18688	74 18944	75 19200	76 19456	77 19712	78 19968	79 20224
50	80 20480	81 20736	82 20992	83 21248	84 21504	85 21760	86 22016	87 22272	88 22528	89 22784	90 23040	91 23296	92 23552	93 23808	94 24064	95 24320
60	96 24576	97 24832	98 25088	99 25344	100 25600	101 25856	102 26112	103 26368	104 26624	105 26880	106 27136	107 27392	108 27648	109 27904	110 28160	111 28416
70	112 28672	113 28928	114 29184	115 29440	116 29696	117 29952	118 30208	119 30464	120 30720	121 30976	122 31232	123 31488	124 31744	125 32000	126 32256	127 32512
80	128 32768	129 33024	130 33280	131 33536	132 33792	133 34048	134 34304	135 34560	136 34816	137 35072	138 35328	139 35584	140 35840	141 36096	142 36352	143 36608
90	144 36864	145 37120	146 37376	147 37632	148 37888	149 38144	150 38400	151 38656	152 38912	153 39168	154 39424	155 39680	156 39936	157 40192	158 40448	159 40704
A0	160 40960	161 41216	162 41472	163 41728	164 41984	165 42240	166 42496	167 42752	168 43008	169 43264	170 43520	171 43776	172 44032	173 44288	174 44544	175 44800
B0	176 45056	177 45312	178 45568	179 45824	180 46080	181 46336	182 46592	183 46848	184 47104	185 47360	186 47616	187 47872	188 48128	189 48384	190 48640	191 48896
C0	192 49152	193 49408	194 49664	195 49920	196 50176	197 50432	198 50688	199 50944	200 51200	201 51456	202 51712	203 51968	204 52224	205 52480	206 52736	207 52992
D0	208 53248	209 53504	210 53760	211 54016	212 54272	213 54528	214 54784	215 55040	216 55296	217 55552	218 55808	219 56064	220 56320	221 56576	222 56832	223 57088
E0	224 57344	225 57600	226 57856	227 58112	228 58368	229 58624	230 58880	231 59136	232 59392	233 59648	234 59904	235 60160	236 60416	237 60672	238 60928	239 61184
F0	240 61440	241 61696	242 61952	243 62208	244 62464	245 62720	246 62976	247 63232	248 63488	249 63744	250 64000	251 64256	252 64512	253 64768	254 65024	255 65280

Oznaczenie wierszy i kolumn oraz górne liczby dziesiętne w poszczególnych polach są identyczne jak w tablicy D.III.2. Dolne liczby dziesiętne powstały w wyniku mnożenia liczb górnych przez 256 (256*0, 256*1, 256*2,...256*255)

Konwersja 2-bajtowej (4-cyfrowej) liczby szesnastkowej na postać dziesiętną polega na dodaniu wartości dziesiętnych dwóch bardziej znaczących cyfr (dolna liczba odpowiedniego pola) i dwóch mniej znaczących cyfr (górna liczba odpowiedniego pola). Przykładowo, wartość liczby szesnastkowej ABCD w systemie dziesiętnym wynosi 43776 + 205 = 43891.

Konwersja liczby dziesiętnej na postać szesnastkową wymaga operacji odejmowania. W pierwszym kroku należy znaleźć wśród dolnych pozycji poszczególnych pól tabeli maksymalną liczbę, nieprzekraczającą wartości danej liczby. Kolumna i wiersz wyznaczają postać szesnastkową dwóch bardziej znaczących cyfr. Znalezioną liczbę dziesiętną odejmujemy od danej liczby, a wynik posłuży do odczytu dwóch mniej znaczących cyfr szesnastkowych.

Dla przykładu wyznaczymy wartość szesnastkową liczby dziesiętnej 45778. Maksymalna liczba, nieprzekraczająca wartości 45778 zapisana w wierszu B0 i kolumnie 2 (B0 + 2 = B2) wynosi 45568. Obliczamy różnicę: 45778 - 45568 = 210. Liczba 210 zapisana jest w wierszu B0 i kolumnie 2 (D0 + 2 = D2). Ostatecznie liczbie dziesiętnej 45778 odpowiada liczba B2D2 w systemie szesnastkowym.

W.Z Graf. W-wa Zam. 9343 K-97