PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW
MERA-PIAP
Al. Jerozolimskie 202 02-222 Warszawa Telefon 23-70-81

Osrodek Automatyki Elektrycznej
Pracownia Oprogramowania Wieloprocesorowych Układów Automatyki i Robotów Przemysłowych

A

Główny wykonawca mgr inż A.Aderek

Wykonawcy mgr inż. M.Porada

071

Konsultant

Nr zlecenia
UR 01 03 01 L

Asembler skrośny dla µP 8086 na mini-komputer SM-4

etap 3 "Opracowanie programu makroprocesora"

Zleceniodawca
Problem węzłowy 06.6

Kierownik Zespołu
dr inż.A.Syryczyński

Pracę rozpoczęto dnia

zakończono dnia 85.09.30

Kierownik Pracowni
mgr inż. A.Aderek

Z-ca Dyrektora ds Automatyki
dr inż. T.Gałązka

Kierownik Ośrodka
prof.dr inż.T.Missala

Praca zawiera:	Rozdzielnik - ilość egz:
stron	Egz. 1 BOINTE
rysunków	Egz. 2 OAE
fotografii	Egz. 3
tabel	Egz. 4
tablic	Egz. 5
załączników tabulogramy	Egz. 6

Nr rejestr. 5475

nie udostępniać wożącznika – poufny

Analiza deskryptorowa komputery, mikroprocesory, programowanie, oprogramowanie skrośne, języki programowania.

Analiza dokumentacyjna

W sprawozdaniu przedstawiono specyfikację zewnętrzną makrogeneratora będącego częścią asemblera skrośnego na minikomputer SM4 dla uP 8086 oraz podano opis języka wejściowego makrogeneratora.

Do sprawozdania dołączono tabulogram programu.

Tytuły poprzednich sprawozdań

Asembler skrośny dla uP 8086 na minikomputer SM-4

etap 1 Uproszczona wersja jednomodułowa

etap 2 Opracowanie założeń na asembler w wersji docelowej.

681.322.001.3 komputery: opis

681.3.066 Programowanie

UKD

MERA-PIAP/TW 331/78 5000

2

OPIS MAKROGENERATORA SKROSNEGO JEZYKA MPL86
NA MASZYNIE SM-4

Spis tresci:

1.WSTEP

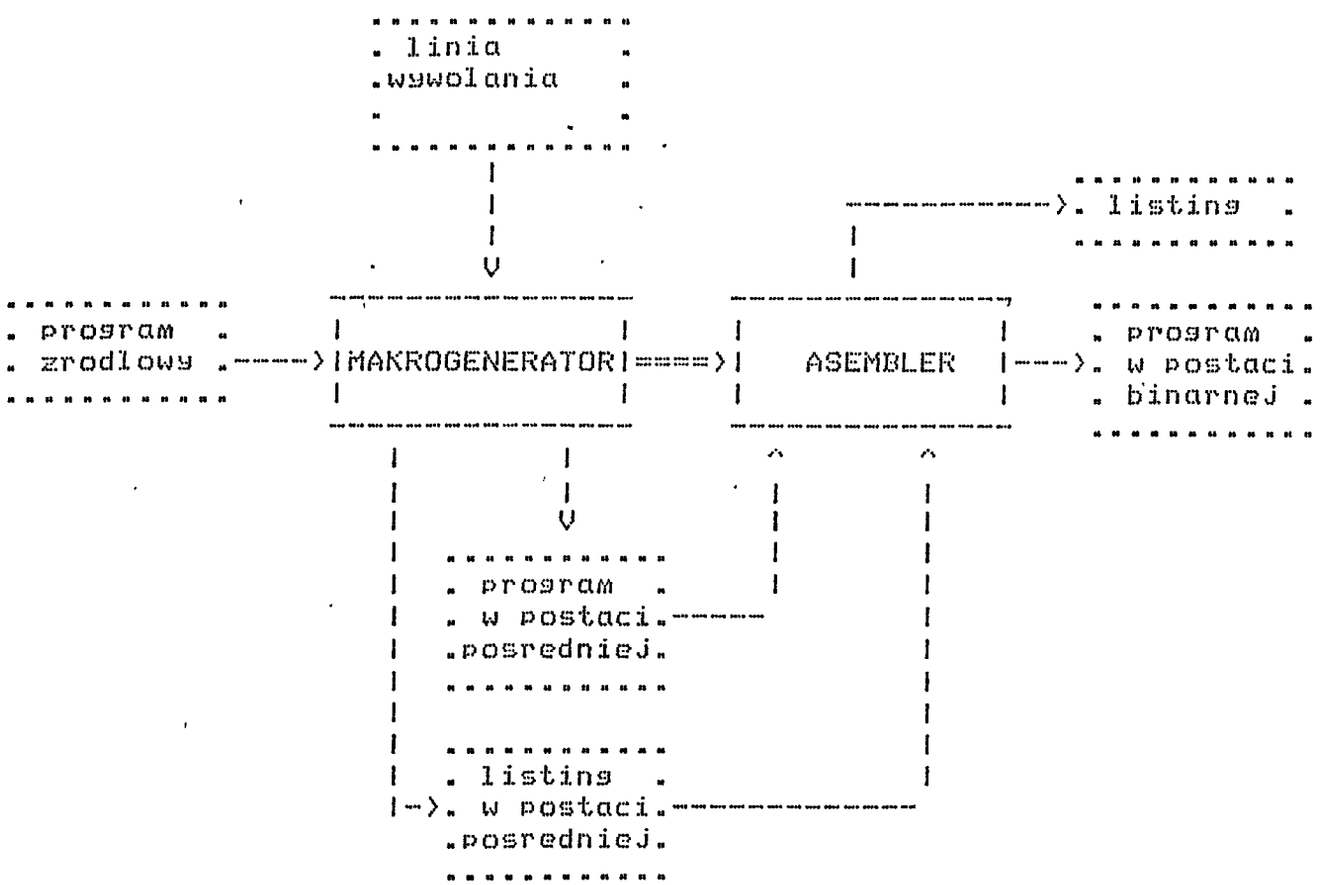
Praca zawiera opis programu makrogeneratora jezyka MPL86, ktory stanowic ma czesc skrosnego makroasemblera dla mikroprocesora 8086 pracujacego na minikomputerze SM-4.Zalozenia na asembler zawarte sa w pracy [1].

Zadaniem makrogeneratora 86 jest przetworzenie tekstu programu zrodlowego,napisanego w jezyku MPL86 do postaci posredniej.Program w tej postaci bedzie dalej kompilowany w kolejnych przebiegach asemblera. Zastosowanie makrogeneratora umozliwi stosowanie podstawien tekstowych w programach napisanych w asemblerze 8086 oraz kompilacje warunkowa tych programow.

2.SPECYFIKACJA ZEWNETRZNA MAKROGENEGATORA

2.1 Miejsce makrogeneratora w strukturze oprogramowania makroasemblera 86

Makrogeneracja zrealizowana jest w odrebnym,pierwszym przebiegu makroasemblera 86. Rozwiazanie takie pozwala na wlaczenie makrogeneratora w istniejace juz oprogramowanie asemblera bez koniecznosci jego przebudowy,oraz zapewnia dobra modularyzacje calego makroasemblera. Przyjecie takiego rozwiazania jest mozliwe ze wzgledu na strukture jezyka MPL86,ktora zapewnia niezaleznosc etapu przetwarzania makrowywolan od etapu asemblacji kodu. Rys.1 przedstawia miejsce makrogeneratora w strukturze oprogramowania makroasemblera.



rys.1

2.2 Wejscia makrogeneratora

Wejsciami makrogeneratora sa:
* program zrodlowy
*linia wywolania

Program zrodlowy

jest programem napisanym w jezyku MPL86 mogacym zawierac ogolne dyrektywy sterujace.Tekst zrodlowy moze byc wprowadzony z dowolnych urzadzen znakowych akceptowanych przez system RSX,z wyjatkiem terminala.

Linia wywolania

podawana jest z terminala-zawiera nazwy plikow i dyrektywy sterujace.(patrz 2.4)

2.3 Wyjscia makrogeneratora

Wyjsciami makrogeneratora sa:
*postac posrednia programu
*postac posrednia listingu

Postac posrednia programu

zawiera przetworzony przez makrogenerator tekst programu zrodlowego.

Postac posrednia listingu

zawiera listing programu zrodlowego utworzony zgodnie z podanymi(w linii wywolania i w tekscie programu zrodlowego)dyrektywami sterujacymi listingiem.

Uwaga:Postac posrednia programu zalezna jest rowniez od dyrektyw GEN NOGEN i GENONLY. Zaleznosc ta wyraza sie tym,ze makrogenerator wprowadza do postaci posredniej puste linie w tych przypadkach,gdy produkowane sa linie w postaci posredniej listingu nie majace odpowiednikow w tekscie postaci posredniej programu.Rozwiazanie takie zapewnia wzajemna odpowiedniosc linii w obu produkowanych zbiorach.

strona 4
stron 16
nr rejestr 5475

2.4 Uruchamianie makrogeneratora

Makrogenerator uruchamiany jest komenda
RUN MPL
Po zgloszeniu sie makrogeneratora nalezy podac
linie wywolania w nastepujacym formacie:

[<objfile>][,<listfile>]=<sourcefile>[lista dyrektyw]
gdzie
<objfile> jest nazwa wynikowego pliku binarnego,
<listfile> jest nazwa pliku na ktorym utworzony
ma byc listing programu,
<sourcefile> jest nazwa pliku zrodlowego.
Tylko podanie sourcefile jest obowiazkowe.
Nazwy plikow musza byc zgodne z formatem obowiazujacym
w systemie RSX.Jezeli ktorykolwiek z plikow nie ma
podanego typu,to przyjmowana jest wartosc
domyslna.
Wartosci domyslne typow plikow sa nastepujace:
dla objfile 086
dla listfile L86
dla sourcefile A86
Kazda dyrektywa w liscie dyrektyw musi byc
poprzedzona znakiem /
W linii wywolania dopuszczone sa nastepujace
dyrektywy:MR,NOMR,GE,NOGE,GO,LI,NOLI.
Uwaga:Pliki objfile i listfile nie sa zapisywane
przez makrogenerator,lecz przez asembler.Makrogenerator
tworzy plik dla postaci posredniej programu
o nazwie ASM86MAC.TMP i plik dla postaci posredniej
listingu ASM86LST.TMP.Oba te pliki sa usuwane w
przebiegach asemblera.Informacje o plikach
objfile i listfile oraz o dyrektywach sterujacych
istotnych w etapie asemblacji sa przekazywane w
pierwszej linii pliku,na ktorym jest postac
posrednia programu.

strona 5
strona 16
ms rejestr 5475

3. OPIS JEZYKA WEJSCIWEGO

Opis jezyka predstawiony jest w oparciu o dokument [2].Tamze znajduja sie przyklady uzycia poszczegolnych jego konstrukcji.
Opis konstrukcji jezyka sklada sie z opisu skladni bezkontekstowej i z opisu semantyki.

Skladnia bezkontekstowa

Jest opisana przy uzyciu notacji BNF nastepujaco rozszerzonej:

* Jednostki syntaktyczne moga byc grupowane przy uzyciu nawiasow klamrowych({i}),wewnatrz ktorych moga wystepowac znaki | oznaczajace alternatywne jednostki skladniowe.
* Powtorzenie grupy jednostek skladniowych dowolna liczbe razy oznaczane jest znakiem +.np {A}+ . oznacza kazdy ciag zlozony z A
* Powtorzenie opcjonalnej grupy jednostek skladniowych dowolna liczbe razy oznaczane jest znakiem *,np{A}* oznacza kazdy ciag zlozony z A i ciag pusty.
* Opcjonalna grupa jednostek skladniowych oznaczana jest przez ujecie jej w nawiasy kwadratowe ([i]).

Opis semantyki

zawiera wyjasnienie znaczenia danej konstrukcji jezyka,a warunki statyczne definiuja warunki zalezne od kontekstu,ktore musza byc spelnione dla poprawnego uzycia danej jednostki skladniowej.

Pojecia ogolne

MAKRO jest fragmentem kodu,ktory ,raz okreslony,moze byc wielokrotnie podstawiany w roznych miejscach programu.DEFINICJA MAKRA okresla sposob w jaki bedzie formulowane zadanie podstawienia-WZORZEC WYWOLANIA oraz tekst,ktory bedzie podstawiany-CIALO MAKRA (wartosc makra).

strona 6
stron 16
urzędzie 5475

1.Teksty

W jezyku wejsciowym wyroznia sie nastepujace
rodzaje tekstow:
Tekst

Jest dowolnym ciagiem znakow ASCII.
Tekst swobodny

Jest dowolnym ciagiem znakow nie zawierajacym metaznaku.
(Metaznak jest znakiem %,o ile nie zostanie
przedefiniowany)
Tekst zrownowazony

Jest dowolnym ciagiem znakow,w ktorym kazdemu
lewemu nawiasowi okraglemu odpowiada prawy nawias
i odwrotnie.Tekst nie zawierajacy nawiasow
okraglych jest rowniez tekstem zrownowazonym.

2.makrowywolanie

składnia:
<makrowywolanie>::=
    <metaznak><funkcja wbudowana>
    |<metaznak><makro wlasne>
semantyka:
Makrowywolanie stanowi zadanie zastapienia
wywolania makra jego wartoscia.Wartoscia makra
jest tekst,ktory zastepuje makrowywolanie.Niektore
makra maja wartosc pusta(tj.tekst zastepujacy
makrowywolanie nie zawiera zadnego znaku).
Metaznak jest znakiem,ktory mozna zdefiniowac
przez wywolanie funkcji wbudowanej
'specyfikacja metaznaku'.Jezeli zaden metaznak nie
zostal zdefiniowany,wowczas przyjmowana jest jego
wartosc domyslna %.
Makro wlasne moze byc wywolane dopiero po jego
zdefiniowaniu.Makro wlasne jest definiowane i
redefiniowane przez wywolanie funkcji wbudowanej
'definicja makro'.
Funkcje wbudowane sa predefiniowane w jezyku i
mosa byc wywolywane bez uprzedniego zdefiniowania.
Funkcje wbudowane nie mosa byc redefiniowane.

strona 7
strau 16
ur rejestr 5475

3.funkcja wbudowana

składnia:

<funkcja wbudowana>::=
    <definicja makro>
  |<komentarz>
  |<przepuszczenie tekstu>
  |<specyfikacja metaznaku>
  |<przekształcenie>
  |<porownanie ciagu znakow>
  |<warunek>
  |<petla warunkowa>
  |<petla iteracyjna>
  |<funkcja wyjscia>
  |<podanie dlugosci ciagu znakow>
  |<czytanie z konsoli>
  |<pisanie na konsole>

4.definicja makro

składnia:

<funkcja definiujaca makro>::=
    *DEFINE<wzorzec makro>
    <lista symboli lokalnych><ciało makro>

<wzorzec makro>::=
   <nazwa makro>
   << wzorzec ogranicznika>
    <lista parametrow formalnych>
    < wzorzec ogranicznika>>

<lista parametrow formalnych>::=
 <<parametr formalny>>
 <<wzorzec ogranicznika><parametr formalny>>*

<parametr formalny>::=
  <identyfikator>

<nazwa makro>::=
   <identyfikator>

<identyfikator>::=
    litera<literal?|_|cyfra dziesietna>*

<lista symboli lokalnych>::=
   <identyfikator>
   < <identyfikator>>*

```
<cialo makro>::=
   <tekst ciala>

<tekst ciala>::=
   {<tekst swobodny>
   |<makrowywolanie>
   |<metaznak><parametr formalny>
   |<metaznak><symbol lokalny>
<wzorzec ogranicznika>::=
   <ogranicznik czysty>
  |<wzorzec ogranicznika literalowego>
<ogranicznik czysty>::=
   <znak czysty>{<znak czysty>}*
<wzorzec ogranicznika literalowego>
  |<znak nie zastrzezony>

<znak czysty>::=
   TAB| |CRLF|LF
```

semantyka:

Wywolanie funkcji definiujacej makro definuje makro wlasne uzytkownika .Wywolanie to definiuje wzorzec wywolania makra(wzorzec makra) oraz wartosc wywolania makra (cialo makra).
Makro jest jednoznacznie identyfikowane przez nazwe okreslona we wzorcu.Powtorne zdefiniowanie makra o tej samej nazwie (redefinicja) powoduje,ze poprzednia definicja przestaje obowiazywac.

Definicja makra musi byc poprzedzona symbolem "*".

Wartoscia wywolania funkcji definiujacej makro jest pusty ciag.

W ciele makra mogą być zawarte makrowywołania .Rozwiniecie ciala makra powoduje zastapienie tych makrowywolan ich wartosciami.

Identyfikatory wymienione w liscie parametrow formalnych mogą występować w ciele makra.Stanowia one znaczniki miejsc w ciele makra,w ktorych maja być podstawione (w czasie rozwijania ciala makra) wartosci parametrow aktualnych.

Identyfikatory wymienione w liscie symboli lokalnych mogą występować w ciele makra.Kazde wywolanie makra powoduje uzupełnienie kazdego z tych identyfikatorow 2-cyfrowym przyrostkiem zawierajacym numer wywołania tego makra.Pierwsze wywolanie danego makra nadaje przyrostek '00H.

warunki statyczne:
* Identyfikatory stanowiące parametry formalne
musza sie roznic pomiedzy soba.Zaden z tych
identyfikatorow nie moze byc nazwa makro.
*znak nie zastrzezony nie moze byc metaznakiem,znakiem
czystym,@,znakiem identyfikatora.
* Tekst ciala musi byc tekstem zrownowazonym.

5.makro wlasne

składnia:
⟨makro wlasne⟩::=
⟨nàzwa makro⟩
{⟨ograniczniK⟩
⟨lista parametrow aktualnych⟩
⟨ograniczniK⟩}

⟨lista parametrow aktualnych⟩::=
⟨parametr aktualny⟩
{⟨ograniczniK⟩⟨parametr aktualny⟩}

⟨parametr aktualny⟩::=
⟨tekst zrownowazony⟩

⟨ograniczniK⟩::=
⟨ograniczniK czysty⟩
|⟨ograniczniK literalowy⟩

⟨ograniczniK literalowy⟩::=
⟨znak nie zastrzezony⟩

semantyka:
Wartoscia wywolania makra wlasnego jest
rozwiniete cialo makra

Teksty bedace parametrami aktualnymi sa rozwijane,nastepnie podstawiane w ciele makra.Dopiero potem rozwijane jest cialo. Parametry aktualne mosa zawierac makrowywolania.

warunki statyczne:

* Liczba parametrow aktualnych musi byc rowna liczbie parametrow formalnych okreslonych we wzorcu wywolania makra o danej nazwie.

*Ograniczniki wystepujace w wywolaniu musza byc zgodne ze swoimi wzorcami w definicji. Jezeli w definicji wystepowal ogranicznik czysty,to w wywolaniu musi tez wystapic ogranicznik czysty.Liczba znakow czystych w definicji i w wywolaniu moze sie roznic i nie musza to byc te same znaki.
Jezeli w definicji wystepowal wzorzec ogranicznika literalowego,to w wywolaniu musi wystapic ogranicznik literalowy bedacy tym samym samym znakiem co we wzorcu.

6.komentarz

składnia:

⟨komentarz⟩::=
'⟨tekst⟩['|LF]

semantyka:

Wartosca funkcji 'komentarz' jest pusty ciag. Funkcja ta pozwala na umieszczanie komentarzy,ktore nie znajda sie w tekscie wyjsciowym,beda natomiast uwidocznione na listingu.

7.przepuszczenie tekstu

składnia:

⟨przepuszczenie tekstu⟩::=
⟨cyfra dziesietna⟩⟨tekst⟩

semantyka:

Wartoscia funkcji 'przepuszczenie tekstu' jest ⟨tekst⟩,ktorego dlugosc okresla ⟨cyfra dziesietna⟩.

warunki statyczne:

* Dlugosc tekstu musi dokladnie odpowiadac wartosci podanej cyfry dziesietnej.

8.specyfikacja metaznaku
składnia:
⟨specyfikacja metaznaku⟩::=
METACHAR(⟨tekst⟩)
semantyka:
Wartoscia funkcji 'specyfikacja metaznaku'jest ciag pusty.Funkcja ta pozwala na redefinicje metaznaku,jakim pierwotnie jest '%'. Nowym metaznakiem staje sie pierwszy znak w tekscie bedącym jej argumentem
warunki statyczne:
* Tekst wystepujacy jako argument tej funkcji musi byc tekstem zrownowazonym.

9.przeksztalcenie
składnia:
⟨przeksztalcenie⟩::=
EVAL(⟨wyrazenie⟩)
semantyka:
Wartoscia funkcji 'przeksztalcenie' jest wartosc wyrazenia ,bedacego argumentem,przedstawiona w postaci szesnastkowej,zgodnie z regulami obowiazujacymi dla reprezentacji liczb szesnastkowych w asemblerze ASM86.Funkcja ta zwraca zawsze conajmniej trzy znaki,z czego pierwszym jest zawsze liczba dziesietna,a nastepne moga byc dowolna liczba w zapisie szesnastkowym.Ostatnim znakiem jest przyrostek 'H'

10.porownanie ciagu znakow
składnia:
⟨porownanie ciagu znakow⟩::=
[EQS|NES|LTS|LES|GTS|GES]
⟨argument1⟩,⟨argument2⟩
⟨argument⟩::=
⟨tekst⟩
semantyka:
Porownanie ciagu znakow realizowane jest przez 6 funkcji wbudowanych (EQS(NES(LTS(LES(GTS(GES).
Funkcje te porownuja(leksykograficznie) dwa teksty
Wartoscia kazdej z tych funkcji jest wartosc logiczna przedstawina w postaci szesnastkowej (00H dla FALSE,-1H dla TRUE).
warunki statyczne:
* Tekst wystepujacy jako argument kazdej z tych funkcji musi byc tekstem zrownowazonym.

strona 12
stron . 16
w rejestr. 5475

11.warunek
skladnia:
<warunek>::=
IF(<wyrazenie>
THEN(tekst)
(ELSE(tekst))
FI
semantyka:
Wartoscia funkcji 'warunek' jest rozwiniety jeden z tekstow bedących jej argumentami,zaleznie od wartosci wyrazenia.Wyrazenie jest obliczane na poczatku.Jezeli najmlodszy bit wartosci wyrazenia jest jedynka,to rozwijany jest tekst1.
Jezeli najmlodszy bit jest zerem,to rozwijany jest tekst2,o ile wystapilo ELSE.W przeciwnym przypadku wartoscia funkcji jest ciag pusty.Funkcje 'warunek' moga byc zagniezdzone.
FI konczy zawsze najglebsze,dotad otwarte wywolanie.
warunki statyczne:
* Tekst wystepujacy jako argument tej funkcji musi byc tekstem zrownowazonym.

12.petla warunkowa
skladnia:
<petla warunkowa>::=
WHILE(<wyrazenie>)
(<tekst>)
semantyka:
Wartoscia funkcji 'petla warunkowa' jest wielokrotnie rozwiniety tekst bedący jej argumentem.
Wartosc wyrazenia obliczana jest na poczatku i po kazdym rozwinieciu tekstu.
Tekst jest rozwijany,gdy najmlodszy bit w wyrazeniu ma wartosc 1.

13.petla iteracyjna
skladnia:
<petla iteracyjna>::=
REPEAT(<wyrazenie>)
(<tekst>)
semantyka:
Wartoscia funkcji 'petla iteracyjna' jest tekst rozwiniety liczbe razy okreslona w wyrazeniu.Wartosc wyrazenia obliczana jest tylko raz,przed wykonaniem pierwszego rozwiniecia.
warunki statyczne:
* Tekst wystepujacy jako argument tej funkcji musi byc tekstem zrownowazonym.

14.funkcja wyjscia
składnia:
⟨funkcja wyjscia⟩::=
EXIT
semantyka:
Wartoscia funkcji wyjscia jest pusty ciag.Funkcja ta konczy rozwijanie ostatnio wywolanego makra wlasnego,funkcji'petla iteracyjna' lub funkcji'petla warunkowa'.

15.podanie dlugosci ciagu znakow
składnia:
⟨podanie dlugosci ciagu znakow⟩::=
LEN(⟨tekst⟩)
semantyka:
Wartoscia tej funkcji jest dlugosc ciagu znakow w rozwinietym tekscie,przedstawiona w postaci szesnastkowej.
warunki statyczne:
* Tekst wystepujacy jako argument tej funkcji musi byc tekstem zrownowazonym.

16.czytanie z konsoli
składnia:
⟨czytanie z konsoli⟩::=
IN
semantyka:
Wartoscia tej funkcji jest jedna linia tekstu przeczytana z konsoli.
Linia ta jest czytana po uprzednim wyslaniu znakow'⟩⟩'.

17.pisanie na konsole
składnia:
⟨pisanie na konsole⟩::=
OUT(⟨tekst⟩)
semantyka:
Wartoscia tej funkcji jest ciag pusty.Funkcja ta wypisuje na konsole tekst bedacy jej argumentem.
warunki statyczne:
* Tekst wystepujacy jako argument tej funkcji musi byc tekstem zrownowazonym.

strona 14
stron 16
nr rejestr 5475

18.wyrazenie

Argumentami wyrazen moga byc liczby binarne(przyrostek B),osemkowe(przyrostek O lub Q),dziesietne(bez przyrostka,lub z przyrostkiem D) i szesnastkowe(przyrostek H).
Dopuszczone sa nastepujace operatory(wymienione w porzadku malejacego pierwszenstwa):
1. ()
2.HIGH,LOW
3.*,/,MOD,SHL,SHR
4.+,-(unarny i binarny)
5.EQ,NE,LE,LT,GE,GT
6.NOT
7.AND
8.OR,XOR
semantyka operatorow jest taka sama,jak w asemblerze 86.
UWAGA: Wewnetrzna reprezentacja liczb w makrogeneratorze jest 16-to bitowa.

Strona 15
strona 16
nr rejestr. 5475

4.DYREKTYWY STERUJACE

Makrogenerator rozpoznaje nastepujace dyrektywy sterujace(w nawiasach podana jest postac skrocona):
MACRO(MR)
NOMACRO(NOMR)
GEN(GE)
NOGEN(NOGE)
GENONLY(GO)
LIST
NOLIST
Dyrektywy MACRO i NOMACRO moga wystapic tylko w linii wywolania.Pozostale dyrektywy moga wystapic zarowno w linii wywolania, jak i w tekscie zrodlowym.
Domyslnie przyjmowane sa nastepujace dyrektywy:
MACRO,GENONLY,LIST.
Znaczenie poszczegolnych dyrektyw jest nastepujace:
MACRO-powoduje,ze makrogenerator bedzie reagowal na metaznak i dokonywal rozwiniec.
NOMACRO-powoduje,ze bedzie realizowany uproszczony przebieg makrogeneratora,tekst zrodlowy bedzie przepuszczony bez zadnych zmian i wyprodukowana postac posrednia listingu.
LIST-powoduje,ze bedzie produkowany tekst listingu
NOLIST-powoduje,ze nie bedzie produkowany tekst listingu.
GEN-powoduje,ze makrogenerator bedzie produkowal listing zawierajacy wszystkie makrowywolania i ich rozwiniecia.
NOGEN-powoduje,ze makrogenerator bedzie produkowal tekst listingu nie zawierajacy rozwiniec(tzn.zawierajacy tylko tekst zrodlowy).
GENONLY-powoduje,ze makrogenerator bedzie produkowal tekst listingu zawierajacy tylko makrorozwiniecia (czyli listing programu w postaci posredniej)

5.LITERATURA

1.A.Aderek,A.Ustaszewski
Asembler skrosny dla mikroprocesora 8086 na minikomputer SM4.OAE-PIAP.1984.

2.8086/8087/8088 Macro Assembly language Reference Manual for 8080/8085 Based Development Systems. Chapter 8.
Intel Corporation 1980.

3.MCS-86 Macro Assembler Operating Instructions for ISIS-II Users.
Intel Corporation 1980.