NASZE GRATULACJE! Masz w swoich rękach jeden z najlepszych i najbardziej uniwersalnych programów użytkowych dostępnych dla Twojego komputera ATARI - kompilator BASIC-a MMG. Jak wiesz, BASIC jest interpreterem. Za każdym razem, gdy uruchamiasz program w tym języku, Twój ATARI przekłada go na język maszyny - zerojedynkowy. W miarę przekładania każdej linii generowany kod maszynowy jest odrzucany, a więc każde ponowne uruchomienie programu powoduje ponowną translację. Rezultaty tego widać w programie, który łatwo jest napisać i poprawić, ale wykonywany jest bardzo wolno. Do tej pory, chcąc mieć szybkie programy, musiałeś nauczyć się programować w języku symbolicznym - assemblerze. Teraz możesz pisać przejrzyste i szybkie programy nie musząc uczyć się nowego języka, tylko korzystając z kompilatora BASIC-a MMG.

Pisz i zmieniaj swoje programy, tak jak to robiłeś zawsze, a potem przełóż je na możliwie najszybszy kod - prawdziwy język maszynowy. Nie jest przy tym wymagana wiedza o tym języku! Kompilator BASIC-a MMG daje możliwość używania liczb całkowitych lub zmienno-przecinkowych, nadaje się więc do zastosowań rozrywkowych i profesjonalnych. W tej chwili na rynku jest kilka gier zręcznościowych napisanych oryginalnie w BASIC-u, a skompilowanych przy pomocy kompilatora BASIC-a MMG.

Kompilator BASIC-a MMG jest 3-przebiegowym kompilatorem generującym prawdziwy kod maszynowy procesora 6502. W przejściu pierwszym zamieniany jest Twój program z BASIC-a na bloki assemblera. W następnych dwóch przejściach

bloki te asemblowane są w kod maszynowy, który następnie jest zapisywany na dyskietce w postaci bloku ładowalnego i uruchamianego z DOS-u lub nazywanego AUTORUN.SYS - uruchamianego automatycznie po włączeniu komputera. Wytworzone bloki w assemblerze mogą być zapisywane na dyskietce. Zaawansowani programiści mogą je wykorzystywać w celu np. łączenia ich z innymi blokami lub odpowiedniego zmieniania.

ROZPOCZĘCIE

Twoja dyskietka z kompilatorem BASIC-a MMG zawiera następujące programy /bloki/:

DOS.SYS
DUP.SYS
AUTORUN.SYS - wyświetla tytuły na ekranie
CMP.OBJ - kompilator BASIC-a MMG używany w pierwszym przebiegu
ASM.OBJ - assembler używany w przebiegach 2 i 3
SYSEQU.ABC - biblioteka systemu równań
SYSLIB.FP - biblioteka zmiennoprzecinkowa
SYSLIB.INT - biblioteka całkowita

Oryginalna dyskietka z kompilatorem BASIC-a MMG jest mechanicznie zabezpieczona przed zapisem. Zabezpieczenia nie należy usuwać, gdyż spowoduje to utratę gwarancji.

Przed rozpoczęciem pracy z kompilatorem MMG sprawdź, czy z komputera usunięte są wszystkie moduły ROM. Załącz stację dyskietek i, gdy zgaśnie jej kontrolna lampka, włóż dyskietkę z MMG i zabezpiecz ją przekręcając uchwyt stacji. Włącz telewizor lub monitor, a następnie Twój komputer ATARI. Właściciele komputerów serii XL muszą przy włączaniu komputera trzymać wciśnięty klawisz OPTION. Po tych czynnościach program zostanie załadowany, na ekranie pojawi się strona tytułowa. Jeśli zapomnisz usunąć wkładki z

BASIC-em lub przycisnąć OPTION /na komputerach XL/, na ekranie pojawi się informacja

"REMOVE CARTRIDGES AND REBOOT" /wyjmij moduły ROM i wczytaj program ponownie/

po której musisz jeszcze raz wczytać program.

Korzystanie z jednej stacji dyskietek

Po pojawieniu się strony tytułowej naciśnij klawisz "D", aby wywołać DOS. Gdy pojawi się menu DOS-u 2.0S, usuń ze stacji dyskietkę z kompilatorem i wsadź nową dyskietkę. Sformatuj ją używając opcji F menu, następnie skopiuj wszystkie bloki systemowe z dyskietki wzorcowej na nową. Dyskietka ta nie będzie działać bez dyskietki wzorcowej, ale będzie użyta podczas kompilacji. Po wykonaniu kopii umieść w stacji dyskietkę wzorcową, wyłącz komputer i wczytaj kompilator od nowa.

Korzystanie z dwóch stacji dyskietek

Po pojawieniu się strony tytułowej naciśnij klawisz "D", aby wywołać DOS, i poczekaj na pojawienie się menu DOS-u 2.0S. Do stacji nr 2 wsuń czystą dyskietkę lub w tym czasie sformatuj inną. Następnie użyj opcji C z menu DOS-u do kopiowania bloków z wzorcowej dyskietki w stacji nr 1 na dyskietkę w stacji nr 2. Nowa dyskietka nie będzie działać bez dyskietki wzorcowej, ale będzie użyta podczas kompilacji. Po skopiowaniu zapisz na dyskietkę z kopią kompilatora program w BASIC-u, który ma być kompilowany i wyłącz komputer. Po tym wczytaj wzorcową dyskietkę z kompilatorem od nowa. Po pokazaniu się na ekranie strony tytułowej zamień dyskietkę wzorcową na inną /czystą/ sformatowaną, a dyskietkę z kopią kompilatora wsadź do stacji nr 2.

KOMPILACJA TWOJEGO PROGRAMU

Zauważ, że istnieje kilka ograniczeń dla Twojego rogramu w BASIC-u /chociaż o wiele mniej niż przy pracy innymi kompilatorami/. Jeśli masz kłopoty z kompilacją ub gdy skompilowany program nie chce działać, to przeczyaj uważnie pozostałą część tej instrukcji, aby rozwiązać aistniały problem.

Pracując z jedną stacją dyskietek będziesz musiał o jakiś czas zamienić dyskietkę zawierającą kopię kompilatora z dyskietką, na której jest Twój program w BASIC-u. Kompilator MMG będzie Cię o to prosił w odpowiednich momentach. W zależności od długosci Twojego programu można uniknąć części zamian przez skopiowanie odpowiedniego bloku SYSLIB /.INT lub .FP/ na dyskietkę zawierającą program w BASIC-u, razem z SYSEQU.ABC i ASM.OBJ. Jednakże, gdy program w BASIC-u jest długi, to na dyskietce może nie być już miejsca na te bloki i bloki assemblera spowodują w tym przypadku wyswietlenie błędu

ERROR 162-DISK FULL /brak miejsca na dyskietce/

i prace będziesz musiał zaczynać od początku. Przy pracy z dwoma stacjami dyskietek wszelkie zamiany dyskietek są wyeliminowane.

Aby załadować kompilator BASIC-a MMG, wciśnij "C" /na stronie tytułowej/. Po załadowaniu komputer poprosi o wsadzenie dyskietki z programem, który ma być skompilowany, oraz o nadanie nazwy programowi. Jeśli chcesz mieć przed odpowiedzią na pytania założony katalog dyskietki w stacji nr 1, to po prostu wciśnij RETURN. W przeciwnym wypadku podaj nazwę kompilowanego programu i wciśnij RETURN. Możesz na przykład podać:

D:MYPROG.BAS lub D2:GAME i RETURN

Następnie zostaniesz zapytany o nazwę bloku wynikowego, który będzie programem w kodzie maszynowym stworzonym dla

Ciebie przez MMG. Dobrą zasadą jest oznaczanie programu w BASIC-u rozszerzeniem .BAS, a kodu wynikowego .OBJ, przy dowolnej ich nazwie. Upewnij się przy tym, że nazwy te różnią się od siebie. Przykłady:

D: MYPROG.OBJ lub D2:GAME.OBJ i RETURN

Jeśli program, który nazwałeś tak jak program w BASIC-u, w rzeczywistości nie jest programem Basicowskim, to kompilator MMG zareaguje:

"FILE NOT BASIC" /program nie w BASIC-u/

i da szansę wprowadzenia innej nazwy.

Po wprowadzeniu obu nazw zostaniesz zapytany, czy arytmetyka ma być prowadzona na liczbach całkowitych czy na zmiennoprzecinkowych. Jeśli chcesz całkowitą, wciśnij "I" i RETURN, a jeśli chcesz zmiennoprzecinkową, to "F" i RETURN. Większa szybkość wykonywania programu jest przy pracy z liczbami całkowitymi, lecz ogranicza to oczywiście arytmetykę w nim użytą. Więcej szczegółów na ten temat znajdziesz w dalszej części tej instrukcji. Po wyborze opcji, całkowitej lub zmiennoprzecinkowej, program zaczyna być kompilowany.

JAK DZIAŁA KOMPILATOR BASIC-a MMG

Pierwszy przebieg

W miarę kompilacji każdej linii Twojego programu na ekranie wyświetlany jest ich numer. W tym samym czasie tworzone są na dyskietce bloki ASSEM.SGA, ASSEM.SGB, ASSEM.SGC itd. Po skończeniu pierwszego przebiegu na ekranie pojawia się informacja

END_OF_PASS1 /koniec pierwszego przebiegu/

a Twój program został przełożony z BASIC-a na assembler.

W tym punkcie mamy dokonaną połowę kompilacji programu w BASIC-u na kod maszynowy.
Pracując z jedną stacją dyskietek na ekranie pojawi się także informacja:

INSERT COMPILER DISC INTO DRIVE 1
THEN PRESS RETURN

/wsuń dyskietkę z kompilatorem do stacji nr 1 i naciśnij RETURN/.

Zrób wszystko zgodnie z poleceniem, aby kontynuować kompilację. Ta oraz inne informacje czy patania pojawiają się wtedy, gdy kompilator LMG nie może znaleźć odpowiedniego bloku. Pracując z dwiema stacjami dyskietek informacje tanie i pytania nie pojawiają się. Nawet gdy korzystasz tylko z jednej stacji, a skopiujesz bloki z dyskietki wzorcowej na dyskietkę zawierającą Twój program w BASIC-u, to też nie musisz zamieniać dyskietek.
Jeśli w czasie pierwszego przebiegu kompilatora zostanie wciśnięty klawisz RESET, to program wróci do początku tego przebiegu.

Drugi i trzeci przebieg

Po kompilacji na assembler LMG ładuje go do komputera, aby przełożyć na kod maszynowy. Operacja ta wymaga dwóch przebiegów, drugiego i trzeciego. Fazy tej konwersji będą wyświetlane na ekranie; w lewym górnym rogu ekranu będzie znajdował się migocący kursor informując, że odbywa się asemblacja. Gdy assembler nie będzie mógł znaleźć potrzebnego bloku, to wyświetlone zostaną informacje i polecenia podobne do przykładowych:

/ASM/ PASS_2	/przebieg 2/
Can't find file →D:ASSEM.SGA	/brak zbioru .../
Please insert Correct Disc	/wsuń właściwą dyskietkę/
PRESS ANY KEY TO CONTINUE	/naciśnij dowolny klawisz/

Po wsunięciu właściwej dyskietki przyciśnij dowolny nie-

funkcyjny klawisz do kontynuowania asemblacji.
Naciśnięcie klawisza RESET w dowolnym momencie po przebiegu pierwszym powoduje przerwanie asemblacji i przywraca sterowanie do DOS-u. Po zakończeniu trzeciego przebiegu Twój program został przełożony na kod maszynowy i zapisany na dyskietce w postaci gotowej do uruchomienia, o pierwotnie nadanej nazwie. Możliwe są teraz do wyboru trzy opcje:

Print line map /pokaż mapę linii/
Run program /uruchom program/
DOS /powrót do DOS/

MAPA ODNIESIENIA

Mapa odniesienia linii jest podprogramem umożliwiającym zainteresowanym programistom dowiedzenie się gdzie się znajduje kod z poszczególnych linii programu BASIC-a. W dodatku liczba 99999 zawiera adres ostatniej komórki pamięci użytej przez skompilowany program, proste jest więc określenie długości otrzymanego kodu maszynowego.

Aby zobaczyć mapę linii, wciśnij "P". Następnie możesz ją oglądać na ekranie, wydrukować lub zapisać na dyskietce. Menu wygląda następująco:

To:

Printer /drukarka/
Screen /ekran/
Disc /dyskietka/

Naciśnij odpowiedni klawisz i RETURN. Pamiętaj, że CTRL-1 zatrzymuje lub odnawia przewijanie ekranu /przy wyborze ekranu/. Po wyborze dyskietki zostaniesz zapytany o nazwę bloku:

DEVICE:FILENAME? → /urządzenie:nazwa bloku?/

Podaj numer stacji i nadaj mapie nazwę, np.

D2:GAME.MAP i RETURN

Mapa zostanie zapisana na dyskietce.

URUCHAMIANIE PROGRAMU

Aby uruchomić nowo skompilowany program, naciśnij "R" i RETURN. Po jego skończonym przebiegu pojawi się informacja:

"BASIC exit" /wyjście z BASIC-a/

oraz:

?Run address> /adres startowy/

Możesz teraz jeszcze raz uruchomić program od dowolnego dziesiętnego adresu, podając ten adres i wciskając RETURN, lub możesz uruchomić bez podawania adresu, wciskając tylko RETURN. Możliwy jest tu także powrót do DOS-u po wpisaniu DOS i wciśnięciu RETURN.

Zauważ, że Twój skompilowany program może być uruchomiony także z DOS-u. Wybierz opcję "L" DOS-u i program automatycznie uruchomi się po załadowaniu do komputera.

Innym rozwiązaniem jest nazwanie skompilowanego programu AUTORUN.SYS i wtedy, gdy na dyskietce jest też DOS, włączenie komputera /gdy w stacji jest ta dyskietka/ spowoduje automatyczne załadowanie i uruchomienie programu.
Do DOS-u możesz przejść od razu po skompilowaniu programu wybierając opcję "DOS" /klawisz "D"/.

UŻYCIE DOS-u

Aby zaoszczędzić czasu oraz liczby zmian dyskietek możesz przenieść kilka podstawowych bloków kompilatora LMG oraz assembler na dyskietkę zawierającą Twój program. Jedynym ograniczeniem jest długość programu w BASIC-u. Bloki, które należy przenieść, to SYSLIB.INT /używając liczb całkowitych/ lub SYSLIB.FP /używając liczb rzeczywistych/ oraz SYSEQU.ABC i assembler ASM.OBJ. Do przeniesienia bloków użyj jednej stacji i skorzystaj z opcji

"O" /DUPLICATE FILE - powielenie bloku/ DOS-u. Przy dwóch stacjach skorzystaj z opcji "C" /COPY FILE - kopiowanie bloku/, wówczas powielenie następuje szybciej.

Ładowanie kompilatora BASIC-a MMG z DOS-u

Kompilator MMG może być załadowany przy pomocy opcji "L" DOS-u. Po pytaniu:

LOAD FROM WHAT FILE? /z którego bloku wczytywać?/

podaj CMP.OBJ i naciśnij RETURN.
Od tego momentu postępuj tak, jak zostało to opisane na początku tej instrukcji /ładowanie kompilatora MMG/.

UWAGI TECHNICZNE

Opcja arytmetyki na liczbach całkowitych

W Twoim programie, napisanym w Atari Basic, wszystkie liczby są w postaci zmiennoprzecinkowej /rzeczywistej/. Dozwolone są więc liczby np. 1, 254, 645, 1.2543, 0.4689. Arytmetyka na liczbach całkowitych umożliwia używanie tylko całkowitych, tzn. całych liczb w zakresie od -32768 do 32767. Dlatego, gdy wybierzesz tę opcję, kompilator MMG wprowadza do kompilowanego programu bibliotekę liczb rzeczywistych całkowitych. Ponieważ arytmetyka liczb całkowitych nie operuje na bardzo dużych lub ułamkowych liczbach, to musisz uważać na niektóre komendy BASIC-a, które nie mogą być przy niej użyte. Są to: COS, SIN, CLOG, LOG, EXP, ATAN; wszystkie one dają w wyniku liczby ułamkowe.

Co więcej, komenda RND/0/ daje liczbę losową z przedziału od 0 do 1. Arytmetyka całkowita nie obejmuje tego zakresu, gdyż można w niej użyć albo 0, albo 1. Aby rozwiązać ten problem, kompilator MMG wprowadza nową funkcję dla liczb losowych. Wprowadź po prostu w nawias przy RND dowolną liczbę /lub zmienną/ całkowitą, a kompilator podaje Ci jako

wynik tej funkcji liczbę losową z przedziału od 0 do podanej liczby minus 1. Aby mieć na przykład liczbę losową z przedziału od 0 do 125, wyrażenie:

X=RND/126/

spowoduje nadanie zmiennej X tej wartości.
Ponieważ arytmetyka całkowita może pracować tylko na liczbach całkowitych, sprawdź czy Twój program w BASIC-u nie zawiera żadnych liczb ułamkowych, w szczególności takich jak: 1.5, 3.14159, 0.25. Chociaż Twój skompilowany program będzie działał, to liczby takie zostaną zamienione na całkowite i wynik działania skompilowanego programu będzie inny od wyniku programu w BASIC-u.
Zwróć uwagę, że pierwiastek kwadratowy z liczby, uzyskany przy pomocy funkcji SQR/X/, daje często w wyniku także liczbę ułamkową. Jeśli będziesz pracował w opcji całkowitej, to wynikiem tej funkcji będzie największa liczba całkowita, której kwadrat jest równy lub mniejszy od liczby pierwotnej /podanej przy SQR/. Nie jest to więc dokładne pierwiastkowanie. W BASIC-u na przykład pierwiastek kwadratowy z 6 wynosi 2.449489...; ta liczba podniesiona do kwadratu jest najbliższa liczbie 6. W opcji całkowitej pierwiastek kwadratowy z 6 wynosi 2, które podniesione do kwadratu daje 4, wynik bardzo różniący się od 6.

Mimo, że arytmetyka całkowita może pracować na liczbach z zakresu od -32768 do 32767, to są wyjątki, w których można używać liczb spoza tego zakresu, mianowicie przy adresowaniu pamięci w instrukcjach PEEK i POKE. Na przykład, chcąc przesuwać poziomo gracza, możesz ciągle zmieniać jego pozycję:

POKE 53248,XPOS

Podobnie - chcąc sprawdzić, czy został wciśnięty jakiś klawisz funkcyjny /OPTION, SELECT lub START/ - możesz ciągle używać instrukcji:

X=PEEK/53279/

Nie możesz jednak poprawnie używać takich instrukcji jak:

PRINT 53279

gdyż spowoduje to wyświetlenie liczby -12257 /tzn. 53279-65536/.

W tym miejscu warto zauważyć, że wszystkie omawiane ograniczenia nie są ograniczeniami kompilatora BASIC-a MMG lecz raczej arytmetyki na liczbach całkowitych. Mogą być one ominięte przez wybór opcji arytmetyki na liczbach rzeczywistych.

Deklaracja zmiennych /COM/

ATARI oferuje dwa typy komend deklarujących zmienne liczbowe i tekstowe. Pierwszą i najbardziej popularną jest DIM. Istnieje także komenda COM, którą używa właśnie kompilator MMG.

Komenda COM jest taka sama jak DIM i jej działanie jest identyczne. Chcąc na przykład zarezerwować miejsce dla 200 znaków zmiennej A$ można zadeklarować:

DIM A$/200/ lub COM A$/200/

Różnica między tymi komendami wychodzi na jaw przy uruchamianiu programu składającego się z kilku zazębionych bloków, tj. gdy pierwszy z bloków ma komendę w rodzaju:

RUN"D:PROGRAM2"

W tym przypadku, oczywiście, wszystkie deklarowane przez DIM zmienne zostają "wyczyszczone" zanim zostanie uruchomiony drugi blok programu.
Czasem jednak zachodzi potrzeba zachowania wartości zmiennych dla następnego programu lub bloku. W takim przypadku po prostu wprowadź te same deklaracje COM w obu programach, a wartości nadane zmiennym w pierwszym programie zostaną zachowane w drugim. Na przykład jeśli chcesz, aby A$ i B zachowały swe wartości w programie drugim, wprowadź

w obu programach następujące linię:

1) COM A3/500/,B/25/

Kompilacja bloków

Posiadając jedną stację dyskietek wszystkie bloki muszą oczywiście znajdować się w stacji nr 1; program będzie prosił o zmianę dyskietek w odpowiednim czasie. Mając dwie stacje kompilator MMG będzie przeszukiwał obydwie stacje, aby znaleźć odpowiedni blok. Jednakże bloki assemblera są zawsze zapisywane w stacji nr 1. Dlatego też, aby maksymalnie wykorzystać objętość dyskietki, Twój program w BASIC-u, bloki wynikowe oraz systemowe powinny znajdować się w stacji nr 2, zostawiając stację nr 1 wyłącznie na bloki assemblera.

Bloki assemblera /ASSEM.SGA, ASSEM.SGB itd./ wymagają mniej więcej 5 razy większej objętości dyskietki niż program w BASIC-u, chociaż kod wynikowy, czyli Twój gotowy skompilowany program, powinien mieć taką samą objętość jak program w BASIC-u /nie licząc biblioteki rozruchowej/. Dlatego też potrzebujesz dyskietki o pięciokrotnie większej ilości wolnych sektorów niż zajmuje program w BASIC-u, tak więc najdłuższe programy, które możesz kompilować używając tylko jednej stacji dyskietek, zajmują około 120 sektorów, chociaż zależy to także od rodzaju programu.

Aby maksymalnie wykorzystać dyskietkę, kompilator MMG posiada opcjonalną komendę powodującą pozostawienie lub kasowanie bloków assemblera w miarę asemblacji programu. Jeśli Twój program w BASIC-u nie zawiera komendy LIST, to bloki assemblera zostaną usunięte w trzecim przebiegu, wtedy gdy zostaje zapisywany Twój kod wynikowy. Inaczej mówiąc, trzeci przebieg robi miejsce na pełnej dyskietce na kod wynikowy usuwając źródłowe bloki assemblera w miarę wykorzystania każdego z nich. Jeśli nie potrzebujesz tego dodatkowego miejsca i chcesz, aby bloki assemblera pozostały, to dołącz po prostu gdziekolwiek w programie w BASIC-u komendę LIST. Została wybrana akurat taka komenda, gdyż nie zajmuje ona

miejsca w języku maszynowym, listowanie takiego programu spowoduje tylko "zaśmieçenie" ekranu zamiast pokazania normalnego kodu BASIC-a. Na końcu programu zalecane jest dopisanie następującej linii:

32767 LIST

Pseudooperatory assemblera

Jeśli chcesz modyfikować bloki assemblera tworzone przez kompilator MMG, powinieneś uważać na następujące komendy rozpoznawane przez assembler /ASM.OBJ/:

.END - kończy asemblację,
.FILE - łączy dwa lub więcej bloków,
= - definiuje symbol,
.BYTE - definiuje bajty danych przechowywanych w pamięci,
.WORD - definiuje adres stałych,
.DBYTE - definiuje ciąg danych w pamięci,
> - definiuje wyższy bajt liczby,
< - definiuje niższy bajt liczby.

BŁĘDY W KOMPILACJI

Zostaną tu wymienione błędy mogące pojawić się podczas kompilacji, zarówno z powodu błędów w programie w BASIC-u oraz błędów systemowych /generowanych przez ATARI/. Różnią się one od błędów mogących pojawić się podczas uruchamiania programu /wszystkie błędy zostaną omówione oddzielnie/.

Błędy systemowe

Wszystkie błędy systemowe są wyświetlane w standardowym formacie ATARI, tj. w postaci liczb. Dokładne informacje o nich znajdziesz w podręczniku o ATARI. Dodatkowo mogą

występować następujące błędy:

SYSTEM ERROR-CAN`T RUN ASSEMBLER /błąd systemowy - nie można uruchomić assemblera/

COMPILATION ABORTED /kompilacja przerwana/

PRESS RETURN TO CONTINUE /naciśnij RETURN klawisz/

Oznacza to, że kompilator MMG napotkał zły blok ASM.OBJ. Problem można rozwiązać przez ponowne skopiowanie wadliwego bloku z dyskietki wzorcowej lub jej kopii.

BAD FREE TEMP /błąd w kompilatorze/

COMPILATION ABORTED /kompilacja przerwana/

PRESS RETURN TO CONTINUE /naciśnij RETURN/

Oznacza to, że istnieje wewnętrzna niekonsekwencja w kompilatorze. Rozwiązaniem jest ponowne załadowanie i uruchomienie kompilatora.

BAD INPUT FROM BASIC FILE /zła dana w bloku BASIC-a/

TOKEN = xxx

COMPILATION ABORTED /kompilacja przerwana/

PRESS RETURN TO CONTINUE /naciśnij RETURN/

Oznacza to, że kompilator MMG napotkał nierozpoznawalny znak w programie w BASIC-u. Przyczyną może być uszkodzona dyskietka. Spróbuj uruchomić program z BASIC-a. Jeśli działa on poprawnie, to wprowadź następujące polecenia:

```
LIST"D:TEMP"
NEW
ENTER"D:TEMP"
SAVE"D:FILENAME"
```

i załaduj kompilator od początku.

Gdy pojawi się dowolny z wymienionych błędów, umieść dyskietkę z blokiem DUP.SYS w stacji nr 1 i naciśnij RETURN, aby oddać kontrolę DOS-owi.

Błędy programowe

Jeśli Twój porgram w BASIC-u ma dowolny z omówionych niżej błędów, to kompilator MMG wyświetli informację o błędzie oraz linie:

SKIPPING STATEMENT /komenda do pominięcia/
CONTINUE OR ABORT /C/A/? /kontynuować czy przerwać?/

Gdy naciśniesz C, to kompilacja będzie kontynuowana pozwalając na wykrycie wszystkich możliwych błędów w programie w BASIC-u. Kompilator MMG ominie po prostu błędne komendy i będzie dalej kontynuował kompilację. Po skompilowaniu całego programu pokaże się informacja:

X ERRORS DETECTED /wykryto X błędów/

jeśli oczywiście jakiekolwiek błędy programowe były.
Program nie może być uruchomiony i kompilacja zostaje w tym miejscu przerwana. Popraw program w BASIC-u, załaduj jeszcze raz kompilator MMG i skompiluj ponownie poprawiony program.
Gdy po pytaniu: CONTINUE OR ABORT /kontynuować czy przerwać/ naciśniesz "A", to powrócisz do menu DOS-u.

Błędy podczas kompilacji

ILLEGALLY PLACED STATEMENT /w złym miejscu umieszczona komenda/

Przyczyna: kompilator napotkał komendę COM po komendzie innej niż COM.
Rozwiązanie: umieść wszystkie komendy COM w liniach o numerach niższych od linii zawierających komendy inne niż COM.

ILLEGAL NEXT /nielegalne NEXT/

Przyczyna: zmienna przy NEXT nie odpowiada zmiennej przy FOR, na przykład FOR I=1 TO 10:NEXT J
Rozwiązanie: popraw zmienne w pętli.

DYNAMIC DIM NOT ALLOWED /nie dozwolona dynamiczna deklaracja w DIM/

Przyczyna: komenda DIM lub COM musi zawierać stałe, a nie zmienne, do określania tablic; nie są dozwolone takie komendy jak np. DIM X/A/ czy DIM AS.

Rozwiązanie: zamień zmienne stałymi.

NEXT WITHOUT FOR /NEXT bez FOR/

Przyczyna: kompilator napotkał komendę NEXT bez odpowiadającego jej FOR.

Rozwiązanie: usuń zbędne NEXT lub wstaw odpowiednie FOR.

RE-DIMENSION ERROR /błąd deklaracji/

Przyczyna: tablica jest deklarowana więcej niż jednokrotnie.

Rozwiązanie: każdą tablicę deklaruj tylko raz.

SYNTAX ERROR /błąd składni/

Przyczyna: w programie w BASIC-u jest błąd w rodzaju brakującej spacji lub przecinka.

Rozwiązanie: popraw program i kompiluj jeszcze raz.

CAN'T COMPILE STATEMENT /niekompilowalna komenda/

Przyczyna: program w BASIC-u zawiera komendę nie rozpoznawaną przez kompilator, w rodzaju LOAD.

Rozwiązanie: usuń takie komendy.

UNDIMENSIONED ARRAY /nie zadeklarowana tablica/

Przyczyna: kompilator napotkał instrukcje zawierające nie zadeklarowane wcześniej tablice /komendami COM lub DIM/.

Rozwiązanie: deklaruj tablice wcześniej niż są używane w programie.

UNDEFINED LINE NUMBERS /nie określone numery linii/

Przyczyna: Podczas przebiegu trzeciego zostały wykryte nie zdefiniowane linie. Jeśli na przykład program w BASIC-u zawiera linie:

```
1ØØ GOTO 1ØØØ
```

a w programie brak linii o numerze 1ØØØ, to assembler zareaguje wyświetleniem nieprawidłowej jego instrukcji wraz z numerem nie zdefiniowanej linii. W tym przypadku na ekranie zostanie wyświetlone:

```
  JMP L1ØØØ
/ASM/ SYSTEM'ERROR
/ASM/ REF:LINE  - 1ØØØ
/ASM/ UNRESOL VED LINE NUMBER
CONTINUE /Y/N/?
```

JMP L1ØØØ jest instrukcją assemblera, numer linii = 1ØØØ. Program pyta, czy chcesz kontynuować kompilację. Kompilując dany program po raz pierwszy powinieneś ją kontynuować i obserwować ewentualne dalsze błędy, naciśnij więc Y. Program po skompilowaniu nie będzie działał z takimi błędami, pamiętaj więc o ich poprawie i ponownej kompilacji poprawionego programu.

GOTO/GOSUB VAR OR EXP /przy GOTO/GOSUB zmienna lub wyrażenie/

Przyczyna: program w BASIC-u zawiera komendy w rodzaju:

```
GOTO A lub GOSUB 1ØØØ+X
```

Rozwiązanie: zmień odpowiednio takie komendy; często możesz je zastąpić liniami:

```
ON A GOTO 1ØØØ,2ØØØ,3ØØØ   lub
ON X GOSUB 1ØØØ,1Ø1Ø,1Ø2Ø,1Ø3Ø
```

Błędy systemowe assemblera:

Poza normalnymi błędami systemowymi na ATARI mogą pojawić

się informacje:

SYSTEM ERROR: 255

Są dwie możliwe przyczyny ich występowania:

1. Nie jest zdefiniowane odniesienie równań bloku, zwykle z powodu uszkodzonego bloku SYSEQU.ABC.
2. Assembler nie może znaleźć następnego bloku kodu źródłowego. Zwykle oznacza to, że bloki były uszkodzone od momentu ich utworzenia w przebiegu pierwszym lub, że był uszkodzony sam kompilator.

Rozwiązanie: Przeprowadź kompilację od nowa po skopiowaniu wszystkich bloków systemowych z dyskietki wzorcowej lub jej kopii.

Błędy podczas uruchamiania

Uruchomienie skompilowanego programu może spowodować pojawienie się dowolnego standardowego błędu ATARI. W skompilowanym programie normalnie może działać komenda TRAP /tak jak w BASIC-u/, która może Ci pomóc w wykrywaniu błędów. Co więcej, PEEK/195/ daje typ wykrytego błędu, także jak w BASIC-u. Jeśli błąd nie jest objęty komendą TRAP lub gdy TRAP jest skasowane, jak np. TRAP40000, to pakiet rozruchowy w skompilowanym programie wypisze adres niewłaściwej instrukcji i pozwoli na wznowienie wykonywania programu od danego adresu po pytaniu:

?Run address> /adres rozruchowy/

Powinien być to dziesiętny adres odpowiadający adresowi linii programu w BASIC-u, jak to pokazano w napie odniesienia linii omawianej wcześniej. Zamiast podania adresu możesz wybrać jeszcze jedną z trzech opcji:

- Wciśnij RETURN dla ponownego uruchomienia programu od początku.
- Wpisz DOS, wciśnij RETURN, aby przywrócić kontrolę do DOS-u

- Naciśnij "C", wciśnij RETURN, aby kontynuować wykonywanie programu od linii, w której wystąpił błąd.

Przykład błędu podczas uruchamiania

Pokazane zostanie, jak określić numer linii, w której jest błąd. Powinieneś wylistować swój program w BASIC-u i skopiować mapę odniesienia linii. Jako przykład rozważmy następujący program:

```
100 REM
110 REM TEST RUN TIME ERROR                    /test na błędy/
120 REM
130 REM PROGRAM WILL GET AN                    /program wykrywa/
140 REM ERROR 11 WHEN I=0                      /błąd 11 gdy I=0/
150 REM
160 FOR I=10 TO 0 STEP -1
170 PRINT 10/I
180 NEXT I
190 END
```

Po uruchomieniu programu z BASIC-a otrzymane zostaną wyniki:

```
1
1.11111111
1.25
1.42857142
1.66666666
2
2.5
3.33333333
5
10

ERROR- 11 IN LINE 170
READY
```

Mapa odniesienia linii tworzy następującą kompilację:

```
LINE   10Ø = 12811
LINE   11Ø = 12811
LINE   12Ø = 12811
LINE   13Ø = 12811
LINE   14Ø = 12811
LINE   15Ø = 12811
LINE   16Ø = 12811
LINE   17Ø = 12825
LINE   18Ø = 12849
LINE   19Ø = 12882
LINE   99999 = 12918
```

Wynika z niej, że skompilowany kod dla linii 17Ø znajduje się w pamięci pomiędzy komórkami 12825 a 12848 /włącznie/. Teraz, gdy uruchomimy skompilowany program, na ekranie zobaczymy:

```
1
1.11111111
1.25
1.42857142
1.66666665
2
2.5
3.33333333
5
1Ø

ERROR- 11
Trace:
1284Ø
?Run address
```

Skompilowany program mówi, że wystąpił błąd 11 i pokazuje ślad adresu, który podaje podprogram wywołujący, który prowadzi do błędu. W tym przypadku nie jest wywoływany żaden podprogram, tak że ślad pokazuje adres 1284Ø. Ponieważ adres

ten zawarty jest pomiędzy adresem początkowym i końcowym dla linii 17Ø, to wiemy stąd, że właśnie w tej linii nastąpił błąd.

TABELA BŁĘDÓW MOGĄCYCH WYSTĄPIĆ PODCZAS URUCHAMIANIA

Numer błędu	Opis błędu
06	brak danych
11	błąd matematyczny /za duża liczba lub dzielenie przez zero/
18	zły znak ciągu tekstowego
128	przerwanie klawiszem BREAK
129	IOCB już otwarty
130	nieistniejące urządzenie
131	IOCB tylko do zapisu
132	zła /nieważna/ komenda
133	urządzenie lub blok programu zamknięte
134	zły numer IOCB
135	błąd czytania z IOCB
136	koniec bloku
137	obcięty zapis
138	przekroczony czas urządzenia
139	NAK urządzenia
140	błąd łącza szeregowego
141	kursor poza zakresem
142	przekroczona szybkość przesyłu łącza szeregowego
143	błąd sprawdzenia sumy łącza szeregowego
144	błąd urządzenia
145	błąd porównania odczytu po zapisie
146	funkcja nie wykonana
147	brak pamięci RAM
160	błąd numeru stacji dyskietek

161	zbyt wiele otwartych bloków
162	dyskietka pełna
163	niepokrywające się dane I/O
164	niepasujący numer bloku
165	błąd numeru bloku
166	błąd długości danych w POINT
167	program /blok/ zablokowany
168	zła /nieważna/ komenda
169	katalog dyskietki pełny
170	blok nie znaleziony
171	zła instrukcja POINT

OPTYMALIZACJA TWOICH PROGRAMÓW

Uwarunkowania czasowe

Wiele programów wymaga petli czasowych, zarówno w celu synchronizacji programu jak i tworzenia w nim przerw. Programy skompilowane działają szybciej; powinieneś zmieniać parametry czasowe korzystając z podanych informacji. Pracując na liczbach rzeczywistych skompilowany program wykonywany jest około 3 razy szybciej od tego samego programu uruchomionego w BASIC-u przed skompilowaniem. Program pracujący z liczbami całkowitymi po skompilowaniu działa około 15 razy szybciej. Najlepszą drogą wykorzystania pętli czasowych jest użycie zegara czasu rzeczywistego ATARI, jak to pokazano w przykładzie.

```
100 DELAY=10:GOSUB 500
110 ...
120 ...
.......
500 TIME=PEEK/20/+DELAY
510 IF PEEK/20/<>TIME THEN GOTO 510
520 RETURN
```

Komórki 18, 19 i 20 są zegarem czasu rzeczywistego na ATARI. Zawartość komórki 20 jest zmieniana po każdej 1/60 sekundy. Gdy komórka 20 zmienia się od 255 do 1, to zawartość komórki 19 w tym okresie zwiększa się o 1 /czyli jednostka na około 4.25 min./. Komórka 18 jest zwiększana po każdym pełnym cyklu komórki 19, czyli o 1 po każdych 1083 min. W podanym przykładzie podprogram w liniach 500-520 korzysta z tych komórek; możliwe jest robienie dowolnych przerw w programie, przy czym będą one takie same w BASIC-u i języku maszynowym, przy arytmetyce rzeczywistej i całkowitej.

JAK ZROBIĆ KRÓTSZE SKOMPILOWANE PROGRAMY

Kompilator BASIC-a MMG został zaprojektowany w celu tworzenia szybkich programów w kodzie maszynowym, ale nie najkrótszych. Większość pętli występujących w Twoim programie zajmuje więcej miejsca po skompilowaniu niż w oryginalnej wersji /w' BASIC-u/. Jedynym typem komendy, która nie zajmuje więcej miejsca, jest GOSUB. Dlatego też używając we wszystkich możliwych miejscach tej komendy /np. zamiast GOTO/ można znacznie zmniejszyć objętość skompilowanego programu. Instrukcje BASIC-a zajmujące dużo miejsca w skompilowanym programie to: obliczenia matematyczne, takie jak np. X+Y+Z; wyrażenia ze zmiennymi tekstowymi, np. A$/I,J/; tablice, jak np. X/I/; pętle FOR - NEXT z krokiem podanym jako zmienna, jak np. FOR I=1 TO 100 STEP B, która zajmuje w skompilowanym programie około 3 razy więcej miejsca od tej samej petli z krokiem np. 2.

UŻYCIE PODWÓJNEJ GĘSTOŚCI

Jeśli posiadasz przynajmniej jedną prawdziwą, o po-

i gęstości zapisu, stację dyskietek, to możesz znacznie zwiększyć rozmiar kompilowanych programów. Program ten nie będzie w ten sposób pracował ze stacją ATARI 1050 /z podwójną gęstością/, lecz jedynie z prawdziwymi stacjami o podwójnym zapisie. Aby pracować z podwójną gęstością, załaduj kompilator tak, jak to zostało opisane wcześniej, i na stronie tytułowej naciśnij "D", aby przejść do DOS-u. Użyj opcji "L" DOS-u do załadowania i uruchomienia programu nazwanego: CHANGE. Program ten zmienia gęstość zapisu stacji. Wykonaj polecenia zawarte w tym programie i obróć wzorcową dyskietkę we wskazanym momencie. Od tej chwili będą oba pracował w podwójnej gęstości i możesz załadować kompilator opcją "L" DOS-u, ładując CMP.OBJ ze strony o podwójnej gęstości dyskietki wzorcowej.

KOMENDY NIE ROZPOZNAWANE PRZEZ KOMPILATOR

Kilka komend BASIC-a z oczywistych przyczyn nie jest rozpoznawanych przez kompilator. Oto one:

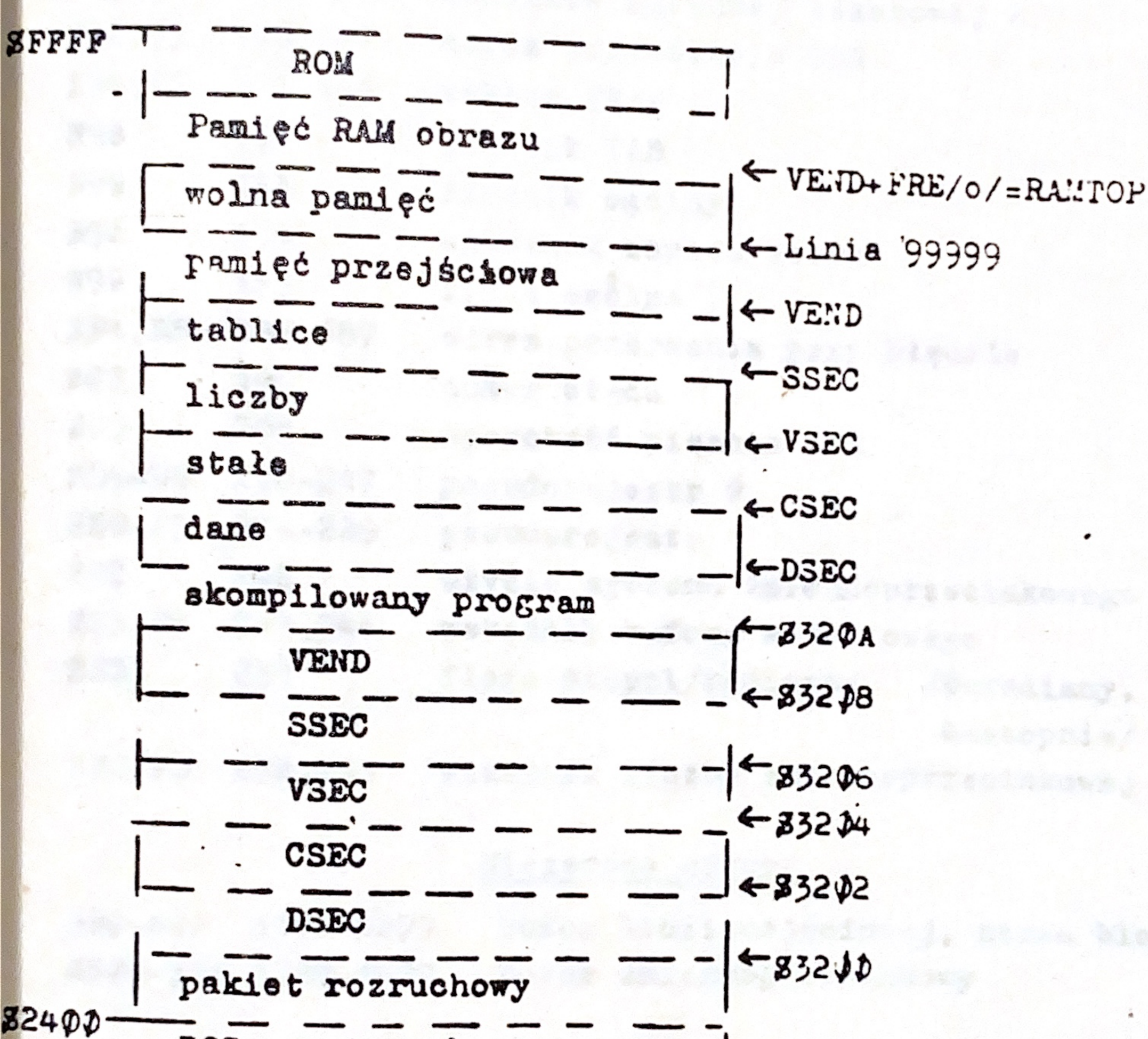
CONT
CLOAD
CSAVE
ENTER
LOAD
NEW
SAVE

Dodatkowo pętle FOR mogą mieć tylko jedną instrukcję NEXT. Po komendach: GOTO, GOSUB i RESTORE nie może być zmienna lecz określony liczbą numer linii.

MAPA PAMIĘCI

System użytkowy ładuje się od $2400, a kod użytkow-

nika rozpoczyna się od $3200 w górę.-Podany diagram pokazuje konfigurację pamięci w czasie uruchamiania skompilowanego programu.



Wejściowy punkt skompilowanego programu jest * $2400.

BIBLIOTEKA PAMIĘCI UŻYTKOWEJ PODCZAS URUCHAMIANIA

Strona zerowa

x.	Dzies.	Opis
0	128	rejestr przestrzeni zapisu
1	129	rejestr przestrzeni zapisu
2,83	130,131	ogólny wskaźnik

$84	132	bieżący kolor punktów /przy PLOT/
$85	133	IOCB dla bieżącego I/O
$86	134	numer komendy dla przywołania z XIO
$88,89	136,137	wskaźnik następnej instrukcji DATA
$8C,8D	140,141	wskaźnik zmiennej tekstowej 1
$90,91	144,145	wskaźnik zmiennej tekstowej 2
$92,93	146,147	adres przywołania USR
$96,97	150,151	wektor TRAP
$98	152	licznik TAB
$99	153	licznik ogólny
$9A	154	wskaźnik zapisu stosu
$9B	155	flaga ogólna
$BA,BB	186,187	adres przerwania przy błędzie
$C3	195	numer błędu
$C9	201	szerokość pisania TAB
$D4-D9	212-217	pseudorejestr 0
$E0-E5	224-229	pseudorejestr 1
$F2	242	użycie systemu zmiennoprzecinkowego
$F3,F4	243,244	wskaźnik bufora wejściowego
$FB	251	flaga stopni/radianów /0=radiany, 6=stopnie/
$FC,FD	252,253	wskaźnik liczby zmiennoprzecinkowej

Niezerowe strony

$B0-4FF	1152-1279	bufor linii wejściowej, nazwa bloku
$500-57F	1280-1407	bufor zmiennoprzecinkowy

WEWNĘTRZNE PRZEDSTAWIANIE LICZB

Format zmiennoprzecinkowy

Liczby zmiennoprzecinkowe /rzeczywiste/ przedstawiane są przy użyciu ATARI OS formatu. Każda liczba rzeczywista przechowywana jest w pamięci w 6 kolejnych bajtach. Znak liczby i przekroczenie 64 przechowywane są w pierwszym bajcie. Następne 5 bajtów zawiera zakodowaną binarnie liczbę dziesiętną; 2 jej cyfry na bajt. Daje to liczbę dziesięciocyfrową.

Format całkowity

Liczby całkowite zajmują 16 bitów i przechowywane są w dwóch kolejnych bajtach. Bajty ułożone są w porządku: najpierw bardziej znaczący bajt, a za nim mniej znaczący. Jest to przeciwne do tego, jak adresuje bajty procesor 6502. Porządek taki został wybrany, aby zapewnić jednolite położenie znaczącego bajtu dla kompilatora i pakietu roz-ruchowego, pozwalając w ten sposób kompilatorowi na two-rzenie kodu niezależnego od opcji arytmetycznej.

KOMERCYJNA SPRZEDAŻ SKOMPILOWANYCH PROGRAMÓW

Żadne władze nie ograniczają sprzedaży programów skompilowanych kompilatorem BASIC-a MMG. My wymagamy jedynie, abyś umieścił w dokumentacji programu następującą uwagę:

> "Program ten został skompilowany przy pomocy kompilatora Basica MMG dla komputera ATARI".

SPIS TREŚCI

WPROWADZENIE 3

ROZPOCZĘCIE 4
Korzystanie z jednej stacji dyskietek 5
Korzystanie z dwóch stacji dyskietek 5

KOMPILACJA TWOJEGO PROGRAMU 6

JAK DZIAŁA KOMPILATOR BASIC-a MMG 7
Pierwszy przebieg 7
Drugi i trzeci przebieg 8

MAPA ODNIESIENIA 9

URUCHAMIANIE PROGRAMU 10

UŻYCIE DOS-u 10
Ładowanie kompilatora BASIC-a MMG z DOS-u 11

UWAGI TECHNICZNE 11
Opcja arytmetyki na liczbach całkowitych 11
Deklaracja zmiennych /COM/ 13
Kompilacja bloków 14
Pseudooperatory assemblera 15

BŁĘDY W KOMPILACJI 15
Błędy systemowe 15
Błędy programowe 17
Błędy podczas kompilacji 17
Błędy podczas uruchamiania 20
Przykład błędu podczas uruchamiania 21

TABELA BŁĘDÓW MOGĄCYCH WYSTĄPIĆ PODCZAS URUCHAMIANIA .. 23

OPTYMALIZACJA TWOICH PROGRAMÓW 24
Uwarunkowania czasowe 24

JAK ZROBIĆ KRÓTSZE SKOMPILOWANE PROGRAMY 25

UŻYCIE PODWÓJNEJ GĘSTOŚCI 25

KOMENDY NIE ROZPOZNAWANE PRZEZ KOMPILATOR 26

MAPA PAMIĘCI 26

BIBLIOTEKA PAMIĘCI UŻYTKOWEJ PODCZAS URUCHAMIANIA 28

WEWNĘTRZNE PRZEDSTAWIANIE LICZB 29
Format zmiennoprzecinkowy 29
Format całkowity 29

KOMERCYJNA SPRZEDAŻ SKOMPILOWANYCH PROGRAMÓW 29