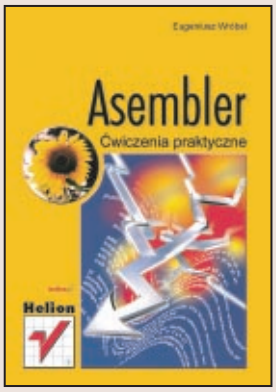
Asembler. Ćwiczenia praktyczne



Autor: Eugeniusz Wróbel
ISBN: 83-7197-836-7
Format: B5, stron: 166

Wydawałoby się, że takie języki jak Java czy C++ całkowicie usunęły asembler w cień. Nie jest to jednak prawda. Fachowcy od asemblera są i będą poszukiwani na rynku pracy. Aplikacje multimedialne, gry, programy działające w czasie rzeczywistym to tylko niektóre obszary wykorzystania tego języka. Dzięki książce czytelnik może dołączyć do elitarnej grupy programistów, którzy potrafią wykorzystać wszystkie możliwości sprzętu i przejąć nad nim pełną kontrolę.

Książka zawiera zestaw ćwiczeń z asemblera procesora Pentium, opracowany przez specjalistów z Politechniki Śląskiej w Gliwicach. Umożliwiają one czytelnikowi zapoznanie się z m.in.:

- Narzędziami używanymi do pisania i uruchamiania programów asemblerowych
- Listą rozkazów procesora Pentium oraz sposobami adresowania argumentów
- Dyrektywami języka MASM
- Sposobami pisania podprogramów i makroinstrukcji
- Metodami obsługi przerwań i pisania programów rezydentnych
- Sposobami optymalizacji kodu

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 3.

Asembler, czyli język zorientowany maszynowo

Jak już wspomniano w rozdziale pierwszym, asembler jest językiem zorientowanym maszynowo i dlatego:

- Program przygotowany dla określonego procesora nie może być wykonany przez inne procesory. W przypadku procesorów serii 80x86 firmy Intel program napisany dla procesorów starszych typów będzie mógł być wykonywany przez procesory nowsze, zachowana jest bowiem (zwana tak w informatycznym żargonie) „kompatybilność" kolejnych procesorów[1].
- Aby programować w asemblerze, trzeba znać podstawową architekturę procesora, w szczególności dostępne programowo rejestry oraz organizację pamięci operacyjnej.
- Aby **dobrze** programować w asemblerze, należy poznać wewnętrzną organizację procesora, pamięci podręcznej, mikroarchitekturę układów dekodujących rozkazy oraz układów wykonawczych i wiele innych niełatwych i czasem trudno dostępnych w literaturze informacji o sposobie wykonywania poszczególnych rozkazów procesora.

[1] Przy pewnych założeniach, np. że nieistotny jest czas wykonania programu.

3.1. Znajomość procesora. Skąd czerpać niezbędną wiedzę o procesorze?

Opis procesorów 80x86 znajdzie Czytelnik w wielu pozycjach literatury, a także w dostępnej na stronach internetowych w postaci plików w formacie *PDF* dokumentacji firmy Intel[2]. Z tego też powodu w niniejszym rozdziale ograniczymy się jedynie do przypomnienia najważniejszych informacji niezbędnych do zrozumienia przykładowych ćwiczeń przedstawionych Czytelnikowi w następnych rozdziałach. Rozpoczniemy od zebrania w tabeli 3.1 niektórych parametrów najważniejszych procesorów serii 80x86.

Tabela 3.1. *Wybrane parametry procesorów 80x86*

Procesor	Data wprowadzenia	Mikro-architektura	Pierwotna częstotliwość zegara	Rozmiar dostępnych rejestrów (w bitach)	Przestrzeń adresowa pamięci	Pamięć podręczna
8086	1978		8 MHz	16	1 MB	—
Intel 286	1982		12,5 MHz	16	16 MB	—
Intel 386 DX	1985		20 MHz	32	4 GB	—
Intel 486 DX	1989		25 MHz	GP: 32 FPU: 80	4 GB	8 KB L1
Pentium	1993	P5	60 MHz	GP: 32 FPU: 80	4 GB	16 KB L1
Pentium Pro	1995	P6	200 MHz	GP: 32 FPU: 80	64 GB	16 KB L1 512 KB L2
Pentium II	1997	P6	266 MHz	GP: 32 FPU: 80 MMX: 64	64 GB	16 KB L1 512 KB L2
Pentium III	1999	P6	700 MHz	GP: 32 FPU: 80 MMX: 64 XMM: 128	64 GB	16 KB L1 256 lub 512 KB L2
Pentium 4	2000	Intel NetBurst micro-architecture	1,4 GHz	GP: 32 FPU: 80 MMX: 64 XMM: 128	64 GB	12 KB μop Execution Trace Cache 8 KB L1 256 KB L2

[2] Nie sposób wyobrazić sobie tworzenia programów asemblerowych bez możliwości ciągłego odwoływania się w czasie pracy do pozycji [10], [11], [12], podanych w bibliografii. Dokumentacja jest udostępniona przez firmę Intel i uaktualniana w miarę pojawiania się nowych procesorów.

Znajomość rodzaju mikroarchitektury, jak i wielkości oraz organizacji pamięci podręcznej potrzebna nam będzie dopiero w przypadku, gdy zaczniemy myśleć o optymalizowaniu naszego programu po względem czasu jego wykonania. Jest to bez wątpienia wyższa szkoła jazdy, o którą delikatnie „otrzemy się", wykonując ćwiczenia w rozdziale 7. Teraz skupimy się na dostępnych programowo rejestrach procesora Pentium, z których korzystać będziemy, programując w asemblerze.

3.2. Podstawowe informacje, potrzebne programiście

Rejestry procesora Pentium pozwalają na zapamiętywanie przetwarzanych danych, służą do sterowania pracą procesora, w tym generowania adresu efektywnego argumentów w pamięci operacyjnej. Warto wiedzieć, iż zestaw podstawowych 32-bitowych rejestrów w procesorze Pentium nawiązuje do 8-bitowych mikroprocesorów firmy Intel — procesorów 8080 oraz 8085, które przed wielu laty zrewolucjonizowały technikę komputerową i pozwoliły „wyjść" komputerom z niedostępnych, tajemniczych i klimatyzowanych hal komputerowych (koniecznie o podwójnych podłogach) na biurko zwykłego śmiertelnika. Sam Intel przyznaje [10], że w procesorze Pentium dopatrzyć można się jeszcze wcześniejszego protoplasty — 4-bitowego pierwszego mikroprocesora 4004 firmy Intel, zaprojektowanego w 1969 roku…

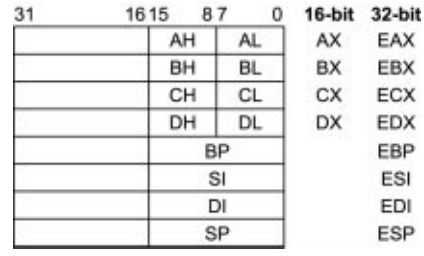
Dla nas istotne jest, że 32-bitowa architektura współczesnych procesorów Pentium, zwana IA-32, zawiera w sobie zarówno procesory 32-bitowe (Intel 386, 486, Pentium), jak i wcześniejsze 16-bitowe (8086, Intel 286).

Podstawowe rejestry procesora

Zaliczymy do nich:

- ❖ osiem rejestrów ogólnego przeznaczenia,
- ❖ sześć rejestrów segmentów (zwyczajowo nazywanych segmentowymi),
- ❖ rejestr znaczników EFLAGS,
- ❖ wskaźnik rozkazów EIP.

32-bitowe rejestry ogólnego przeznaczenia EAX, EBX, ECX, EDX, EBP, ESI, EDI oraz ESP, pokazane na rysunku 3.1, służą do operowania na argumentach operacji arytmetycznych i logicznych, argumentach służących do obliczania adresu efektywnego oraz na wskaźnikach pamięci. Szczególną rolę pełni rejestr ESP, który jest wskaźnikiem szczytu stosu.



Rysunek 3.1.
Podstawowe rejestry procesora Pentium

Młodsze 16 bitów tych ośmiu rejestrów to odpowiedniki rejestrów ogólnego przeznaczenia procesorów 8086 oraz Intel 286, dostępne odpowiednio poprzez nazwy: AX, BX, BC, DX, BP, SI, DI oraz SP. Każdy z dwóch bajtów rejestrów AX, BX, CX oraz DX dostępny jest niezależnie poprzez nazwy: AH, BH, CH, DH (starsze bajty — *High*) oraz AL, BL, CL, DL (młodsze bajty — *Low*). Te ostatnie cztery rejestry nawiązują bezpośrednio do wspomnianego już procesora 8080. Nazwy rejestrów pochodzą od głównego ich przeznaczenia. I tak:

- *EAX* — akumulator, główny rejestr do przechowywania argumentów oraz wyników obliczeń;
- *EBX* — rejestr bazowy, wskaźnik do danych w segmencie, zdefiniowanym przez rejestr DS;
- *ECX* — licznik pętli oraz operacji na łańcuchach;
- *EDX* — wskaźnik układów wejścia-wyjścia;
- *ESI* — rejestr indeksowy, wskaźnik do danych w segmencie, zdefiniowanym przez rejestr DS, wskaźnik argumentu źródłowego dla operacji na łańcuchach;
- *EDI* — rejestr indeksowy, wskaźnik do danych w segmencie, zdefiniowanym przez rejestr ES, wskaźnik argumentu przeznaczenia dla operacji na łańcuchach;
- *ESP* — wskaźnik stosu w segmencie, zdefiniowanym przez rejestr SS;
- *EBP* — rejestr bazowy, wskaźnik do danych w segmencie stosu, zdefiniowanym przez rejestr SS.

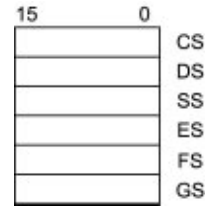
W procesorze Pentium, pracującym w trybie 32-bitowym, możliwe jest bardzo elastyczne używanie wymienionych powyżej rejestrów, czasem w inny sposób, aniżeli wynikałoby to z ich nazwy i zastosowania w przypadku pracy w trybie 16-bitowym. Na przykład rejestry indeksowe mogą pełnić rolę rejestrów bazowych i odwrotnie, rolę rejestrów indeksowych lub bazowych pełnić mogą także pozostałe rejestry 32-bitowe.

Kolejną grupę rejestrów stanowią 16-bitowe rejestry segmentowe: CS, DS, SS, ES, FS, GS. Rejestry te odgrywają istotną rolę w organizacji pamięci operacyjnej. W przypadku pracy procesora w trybie adresacji rzeczywistej rejestry segmentowe zawierają 16 starszych bitów 20-bitowego adresu początku segmentów. W trybie adresacji wirtualnej z ochroną rejestry zawierają selektory segmentów, czyli informacje, które w sposób nieco bardziej złożony niż w trybie adresacji rzeczywistej pozwalają zdefiniować segmenty w pamięci operacyjnej. Ponieważ w książce, którą Czytelnik ma w ręce, nie będziemy zajmować się konstrukcją systemów operacyjnych, a ćwiczenia ograniczymy zasadniczo do przykładów programów pracujących w trybie adresacji rzeczywistej, zagadnienia selektorów nie będziemy tutaj bardziej szczegółowo opisywać.

Poszczególne rejestry definiują (rysunek 3.2):

- *CS* — segment programu;
- *SS* — segment stosu;
- *DS* — podstawowy segment danych;
- *ES*, *FS*, *GS* — dodatkowe segmenty danych.

Rysunek 3.2.
Rejestry segmentowe



Kolejny rejestr tej grupy to 32-bitowy rejestr EFLAGS, który zawiera następujące znaczniki[3]:

- stanu, określające stan procesora po wykonaniu operacji arytmetycznej bądź logicznej,
- sterujące, pozwalające wprowadzić procesor w określony stan,
- systemowe.

Rysunek 3.3, zaczerpnięty z [10], przedstawia rozmieszczenie w rejestrze EFLAGS poszczególnych znaczników. Warto zauważyć, że młodsze 16 bitów tego rejestru to rejestr znaczników FLAGS procesora Intel 286, zaś najmłodszy bajt zawiera znaczniki znane już ze wspomnianego procesora 8080.

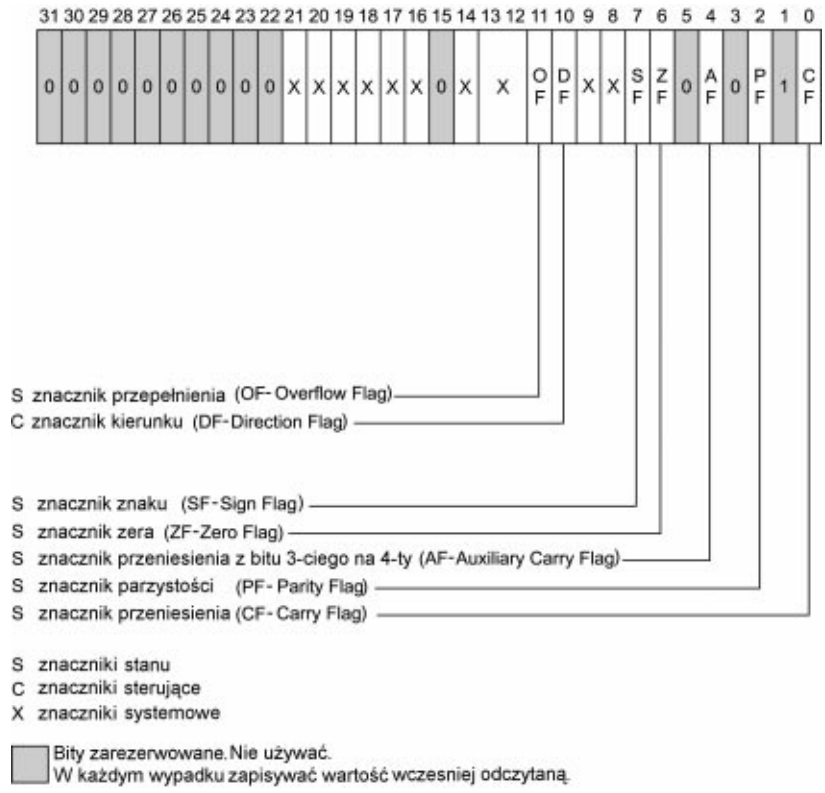
Ostatnim rejestrem tej grupy jest 32-bitowy wskaźnik rozkazów EIP (zawierający w sobie 16-bitowy wskaźnik IP procesorów 8086 i 80286). Rejestr ten wskazuje adres względem początku segmentu programu, skąd pobierany będzie kolejny bajt rozkazu. Rejestr EIP (IP) przeładowywany jest w momencie przekazywania sterowania do innego miejsca w programie, czyli w momencie wykonywania skoku. Jego zawartość będzie dla nas interesująca głównie w czasie śledzenia wykonywania programu przy użyciu debugera.

Rejestry koprocesora arytmetycznego

Architektura procesorów 8086, Intel 206 oraz Intel 386 pozwalała na współpracę z maksymalnie ośmioma koprocesorami, które w znacznym stopniu mogły pracować równolegle z procesorem głównym. Największe zastosowanie miał koprocesor arytmetyczny, oznaczony odpowiednio jako 8087, 80287 oraz 80387. Począwszy od procesora Intel 486 koprocesor arytmetyczny stał się integralną częścią procesora głównego (zatem nie występuje już jako odrębny układ scalony). Z punktu widzenia programisty fakt ten nie

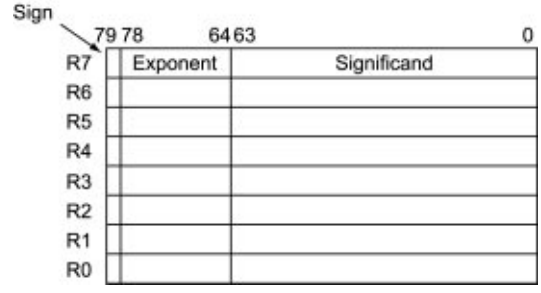
[3] Nazywane przez niektórych programistów żargonowo „flagami”.

Rysunek 3.3. *Rozmieszczenie znaczników stanu i znaczników sterujących w rejestrze EFLAGS*



ma istotnego znaczenia, bowiem architektura tej części procesora Pentium, która realizuje funkcje koprocesora arytmetycznego występującego dawniej jako oddzielny układ scalony, widziana jest przez programistę tak samo jak dawniej — praktycznie bez zmian. Koprocesor arytmetyczny pozwala wykonywać operacje na liczbach zmiennoprzecinkowych. Wykorzystywane są tutaj nowe, 80-bitowe rejestry, zorganizowane w 8-elementowy stos (rysunek 3.4).

Rysunek 3.4. *Rejestry danych koprocesora arytmetycznego*

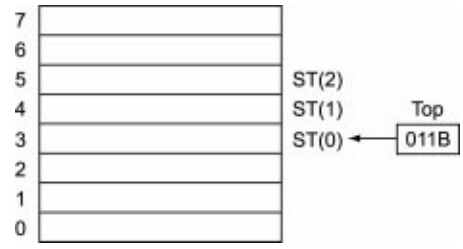


W programie używać będziemy następujących nazw tych rejestrów:

- *ST* lub *ST(0)* — jeden z rejestrów R0-R7, aktualnie wskazywany przez 3-bitowy wskaźnik szczytu stosu, znajdujący się w rejestrze sterującym koprocesora;
- *ST(1)*, ..., *ST(7)* — kolejne rejestry stosu, liczone od aktualnego rejestru ST(0).

Zależności między rejestrami R0 – R7 a rejestrami ST, które stosujemy w programie, ilustruje rysunek 3.5.

Rysunek 3.5.
Rejestry koprocesora, traktowane jako stos



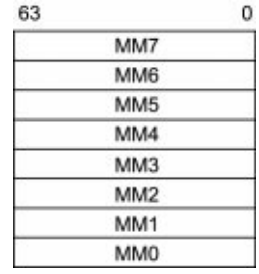
Pewna liczba rozkazów operujących na rejestrach koprocesora arytmetycznego powoduje automatyczne zwiększenie wskaźnika stosu o 1 lub 2. Stos „zawija się", tzn. na powyższym rysunku rejestr R7 w naszym programie będzie występował jako ST(4), natomiast R0 jako ST(5).

Rejestry rozszerzenia MMX

Rozszerzenie MMX pozwoliło zwiększyć możliwości procesora Pentium w zakresie przetwarzania obrazu oraz dźwięku. Jest to klasyczny przykład tzw. architektury SIMD (*Single Instruction Multiple Data Execution Mode*). Programista otrzymuje do dyspozycji dodatkowych 8 rejestrów 64-bitowych, nazywanych MM0, …, MM7 oraz nowe typy danych 64-bitowych (por. rysunek 3.6):

- ❖ 8 spakowanych bajtów,
- ❖ 4 spakowane słowa 16-bitowe,
- ❖ 2 spakowane podwójne słowa 32-bitowe,
- ❖ 1 słowo 64-bitowe.

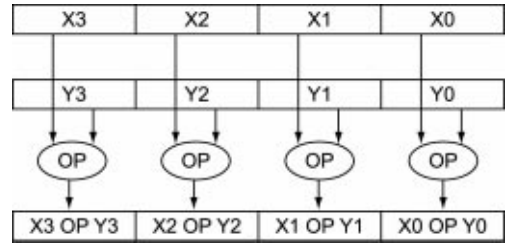
Rysunek 3.6.
Rejestry rozszerzenia MMX



Słowo „spakowane" należy w tym miejscy rozumieć jako umieszczone obok siebie w jednej, 64-bitowej danej.

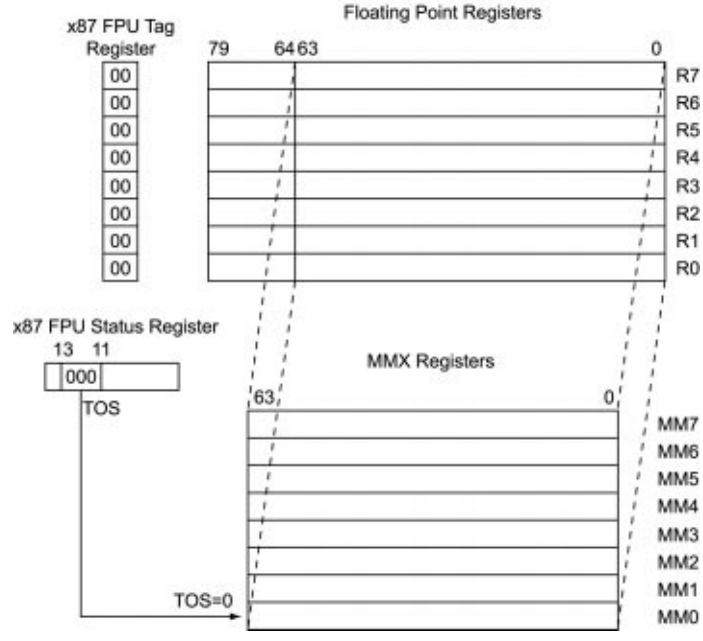
Możliwe jest równoległe wykonywanie operacji arytmetycznych na całych danych 64-bitowych, traktowanych odpowiednio jako bajty, słowa bądź podwójne słowa. Przykładowo ilustruje to rysunek 3.7.

Rysunek 3.7.
Typowa operacja, wykonywana na rejestrach MMX



Należy pamiętać, że w rzeczywistości rejestry MMX są „mapowane" na rejestrach koprocesora arytmetycznego i dlatego nie jest możliwe mieszanie rozkazów, wykorzystujących koprocesor arytmetyczny oraz wykorzystujących technologię MMX. Ilustruje to rysunek 3.8, zaczerpnięty z [12].

Rysunek 3.8.
Odwzorowanie rejestrów MMX na rejestrach koprocesora arytmetycznego

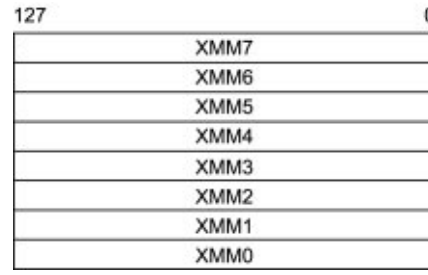


Rejestry MMX w przeciwieństwie do rejestrów koprocesora dostępne są wprost poprzez swoje nazwy: MM0, MM1, …, MM7.

Rejestry rozszerzenia SSE oraz SSE2

Rozszerzenie SSE, jakie pojawiło się z chwilą wprowadzenia procesora Pentium III oraz jego dalsze rozwinięcie w Pentium 4, występujące pod nazwą SSE2, jest w pewnym stopniu połączeniem idei koprocesora arytmetycznego, operującego na liczbach zmiennoprzecinkowych, oraz rozszerzenia MMX, które pozwala równolegle (równocześnie) przetwarzać kilka danych, spakowanych w jednej danej. Programista otrzymuje do dyspozycji osiem nowych rejestrów 128-bitowych, dostępnych bezpośrednio poprzez nazwy XMM0, XMM1, …, XMM7 (por. rysunek 3.9).

Rysunek 3.9. *128-bitowe rejestry rozszerzenia SSE oraz SSE2*



Rozszerzenie SSE wprowadza nowy format spakowanej 128-bitowej danej, zawierającej cztery 32-bitowe liczby zmiennoprzecinkowe pojedynczej precyzji. Rozszerzenie SSE2 rozszerza te możliwości o kolejne formaty. Programista, korzystając z rozkazów operujących na danych typu SSE oraz SSE2, ma w Pentium 4 do dyspozycji następujące formaty danych 128-bitowych:

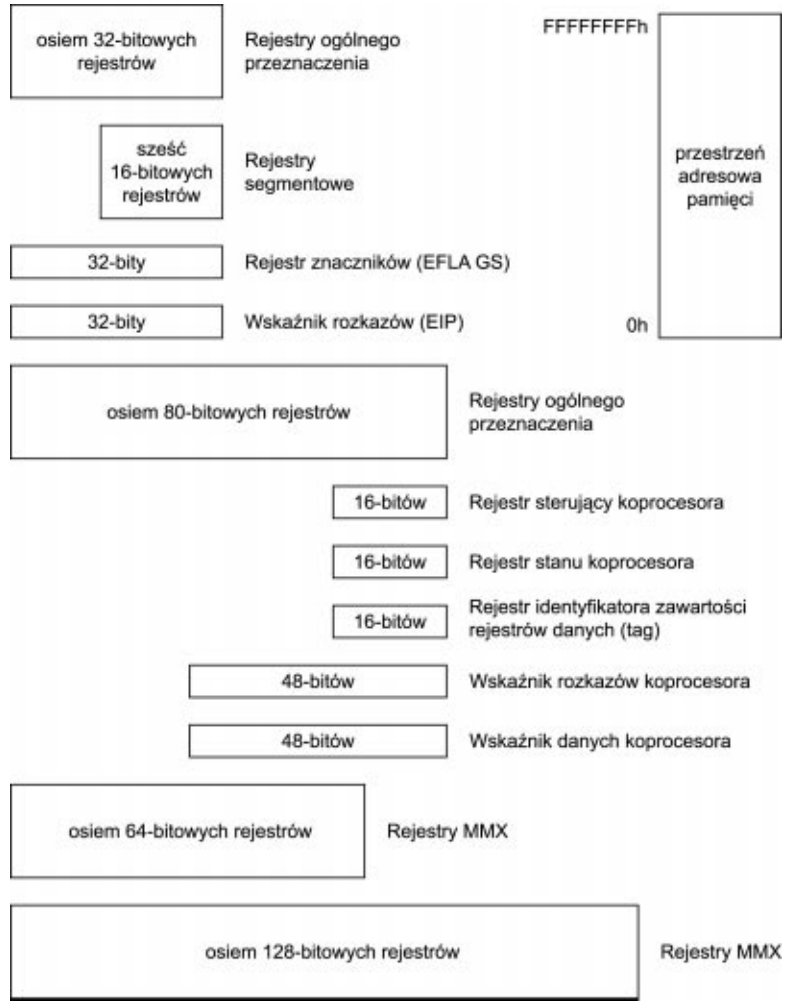
1. Wartości zmiennoprzecinkowe:
   - ❖ 4 wartości zmiennoprzecinkowe pojedynczej precyzji[4],
   - ❖ 2 wartości zmiennoprzecinkowe podwójnej precyzji.
2. Wartości całkowite:
   - ❖ 16 bajtów,
   - ❖ 8 słów,
   - ❖ 4 podwójne słowa,
   - ❖ 2 poczwórne słowa.

Na koniec tego pobieżnego przypomnienia elementów architektury, niezbędnego dla zrozumienia ćwiczeń omówionych w kolejnych rozdziałach, zbierzemy na jednym rysunku (rysunek 3.10) środowisko, w jakim porusza się programista uruchamiający program na procesorze Pentium 4.

3.3. Skąd czerpać wiedzę na temat asemblera?

Jak już kilkakrotnie wspomniano w tej książce, dostępność literatury na temat języka asemblera procesorów Pentium nie jest zbyt duża. Wystarczy wejść do dowolnej księgarni technicznej, aby na półkach znaleźć wiele pozycji dotyczących baz danych, języka C, Pascala i być może jedną pozycję dotyczącą asemblera. Najczęściej jednak nie znajdziemy żadnej pozycji. Sięgnięcie do największych światowych księgarń internetowych, takich jak np. *www.amazon.com*, także nie daje satysfakcjonujących rezultatów.

[4] Występuje także w rozszerzeniu SSE.



Rysunek 3.10. *Podstawowe środowisko programisty w języku asemblera*

W chwili pisania tych słów jedynymi w miarę aktualnymi pozycjami, dostępnymi na stronach tej księgarni, były książki [7] i [8], które nie są jednak pełnym opisem asemblera, pozwalają jedynie (trzeba przyznać, że dosyć gruntownie) poznać podstawy tego języka. Pozostaje oczywiście niezawodny Internet, w którym doświadczony internauta znajdzie prawie wszystko.

Wiemy już, że aby programować w asemblerze, musimy znać:

- architekturę procesora i jego listę rozkazów, a także organizację pamięci operacyjnej — tę wiedzę skutecznie uzyskamy na stronach *www.intel.com*[5],
- opis języka MASM[6], strukturę programu asemblerowego, dyrektywy, pseudoinstrukcje, makroinstrukcje asemblera.

[5] Na potrzeby tej książki korzystaliśmy z dokumentacji udostępnionej na stronach producenta procesorów, w szczególności z pozycji [1], [2], [3], [10], [11] i [12].

Najcenniejszą pozycją drukowaną, opisującą język MASM jest [9] — niestety obecnie praktycznie jest ona nieosiągalna. Niewiele też niestety można znaleźć na stronach *www.microsoft.com*. Pozostaje szukanie w sieci na stronach przede wszystkim uczelni wyższych, na których prowadzone są wykłady z tego zakresu oraz na stronach różnych pasjonatów asemblera. Na tych ostatnich można znaleźć wiele ciekawych informacji, przykładowych programów, a nawet krótkich kursów programowania w asemblerze. Nie będziemy w tym miejscu podawać konkretnych adresów stron WWW, bowiem ulegają one zmianom i szybko się dezaktualizują.

Warto także śledzić dyskusję na grupach dyskusyjnych, poświęconych asemblerowi procesorów 80x86[7].

[6] W książce opieramy się na języku makroasemblera MASM firmy Microsoft. Dla procesorów Pentium dostępne są także inne asemblery, w szczególności TASM firmy Borland, mający wiele identycznych bądź podobnych do języka MASM elementów oraz NASM — o zupełnie innej filozofii niż MASM i TASM. Język NASM dostępny jest w Internecie na podobnych zasadach, jak system operacyjny Linux.

[7] Dostępne grupy w chwili pisania książki: *comp.lang.asm.x86* (angielskojęzyczna), *de.comp.lang.assembler.x86* (niemieckojęzyczna).