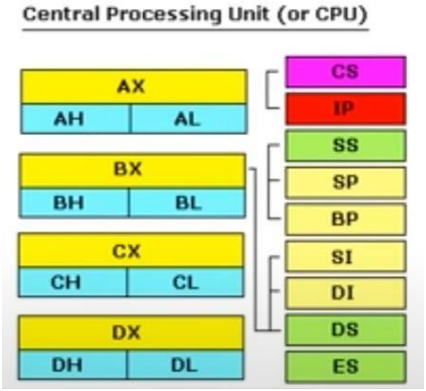
Archi, Assembly, Dziwki, Drugi, Lasery, Sos, Ciuchy i Borciuchy

Autor: puckmoment na discordzie (podziękowania jeśli się przydało mile widziane)

Motywacja od turasa

Czy nauczenie się tego jest trudne?

Rejestry



General purpose registers

Procesory 8086 posiadają 8 16bitowych rejestrów ogólnego przeznaczenia:

- AX (akumulator) – Jest to rejestr akumulatorów. Jest używany w instrukcjach arytmetycznych, logicznych i przesyłania danych. W manipulacji i dzieleniu jedna z liczb musi być w AX lub AL.
- BX (rejestr podstawowy) – Jest to rejestr podstawowy. Rejestr BX jest rejestrem adresowym. Zwykle zawiera wskaźnik danych używany do adresowania pośredniego opartego, indeksowanego lub rejestrowego.
- CX (rejestr liczników) – Jest to rejestr liczników. Służy jako licznik pętli. Ułatwia to konstrukcje pętli programu. Rejestr liczników może być również używany jako licznik w manipulacji ciągami i instrukcjach shift/rotate.
- DX (rejestr danych) – Jest to rejestr danych. Rejestr danych może być używany jako numer portu w operacjach wejścia/wyjścia. Jest również używany w mnożeniu i dzieleniu.
- SP (wskaźnik stosu) – Jest to rejestr wskaźnika stosu wskazujący na stos programu. Jest używany w połączeniu z SS w celu uzyskania dostępu do segmentu stosu
- BP (wskaźnik bazowy) – Jest to rejestr wskaźnika bazowego wskazujący na dane w segmencie stosu. W przeciwieństwie do SP, możemy użyć BP, aby uzyskać dostęp do danych w innych segmentach.

- SI (indeks źródłowy) – Jest to rejestr indeksu źródłowego, który jest używany do wskazywania lokalizacji pamięci w segmencie danych adresowanym przez DS. Tak więc, gdy zwiększamy zawartość SI, możemy łatwo uzyskać dostęp do kolejnych lokalizacji pamięci.
- DI (indeks przeznaczenia) – Jest to rejestr indeksu przeznaczenia pełni tę samą funkcję, co SI. Istnieje klasa instrukcji zwanych operacjami na ciągach znaków, które wykorzystują DI w celu uzyskania dostępu do lokalizacji pamięci adresowanych przez ES.

Segment registers

- Segment kodu (CS) – Segment kodu (CS) to 16-bitowy rejestr zawierający adres segmentu o rozmiarze 64 KB z instrukcjami procesora. Procesor wykorzystuje segment CS dla wszystkich dostępów do instrukcji, do których odwołuje się rejestr wskaźnika instrukcji (IP). Rejestr CS nie może być zmieniony bezpośrednio. Rejestr CS jest automatycznie aktualizowany podczas instrukcji dalekiego skoku, dalekiego wywołania i dalekiego powrotu.
- Segment stosu (SS) – Segment stosu to 16-bitowy rejestr zawierający adres segmentu o rozmiarze 64KB ze stosem programu. Domyślnie preprocesor zakłada, że wszystkie dane, do których odwołują się rejestry wskaźnika stosu (SP) i wskaźnika bazowego (BP), znajdują się w segmencie stosu. Rejestr SS można zmienić bezpośrednio za pomocą instrukcji POP / PUSH.
- Segment danych (DS) – Segment danych to 16-bitowy rejestr zawierający adres segmentu o rozmiarze 64KB z danymi programu. Domyślnie procesor zakłada, że wszystkie dane, do których odwołują się rejestry ogólne (AX, BX, CX, DX) i rejestr indeksowy (SI, DI) znajdują się w segmencie danych. Rejestr DS można zmienić bezpośrednio za pomocą instrukcji POP / PUSH i LDS.
- Dodatkowy segment (ES) – Segment dodatkowy to 16-bitowy rejestr zawierający adres segmentu o rozmiarze 64KB, zwykle z danymi programu. Domyślnie procesor zakłada, że rejestr DI odwołuje się do segmentu ES w instrukcjach manipulacji ciągami. Rejestr ES można zmienić bezpośrednio za pomocą instrukcji POP / PUSH i LES.
- IP (wskaźnik instrukcji) – Jest to specjalny segment dostępny w mikroprocesorze 8086. Rejestr CS zawiera numer segmentu następnej instrukcji, a IP zawiera offset. Adres IP jest aktualizowany za każdym razem, gdy wykonywana jest instrukcja, tak aby wskazywał na następną instrukcję. W przeciwieństwie do innych rejestrów, IP nie może być bezpośrednio manipulowane przez instrukcję, to znaczy, instrukcja nie może zawierać IP jako swojego argumentu.

Deklaracja typów i zmiennych

Typy danych

- DB (define byte) – jeden bajt
- DW (define word) – 2 bajty
- DD (define doubleword) – 2 bajty * 2, czyli 4 bajty
- DQ (define quadword) – 2 bajty * 4, czyli 8 bajtów

Deklarowanie zmiennych

Przypisanie wartości 20 do zmiennych a i b:

```
;Nazwa          Typ     Wartość
a               DB      20
```

```
b               DB      14h
```

Przypisywanie tekstu:

```
;Nazwa          Typ     Wartość
slowo           DB      "Witam :D",13,10,"$"
```

Każda oddzielna litera ma typ zmienniej DB.

13,10 jest odpowiedzialne za znak nowej linii

`"$"` to znak końca stringa

Tablice

Deklarowanie tablicy

```
;Nazwa          Typ     Wartości
Tablica         DB      01h, 02h, 00h, 10h, 12h, 33h
```

Dostęp do danych z tablicy można uzyskać na 2 sposoby:

Przez najprostsze w świecie wpisanie konkretnej wartości:

```
mov     al, Tablica[1] ; wartość 02h trafi do rejestru al.
```

Lub poprzez rejestr SI (Source index / indeks źródłowy):

```
mov     si, 2
mov     al, Tablica[si] ; wartość 00h trafi do rejestru al
```

Instrukcje

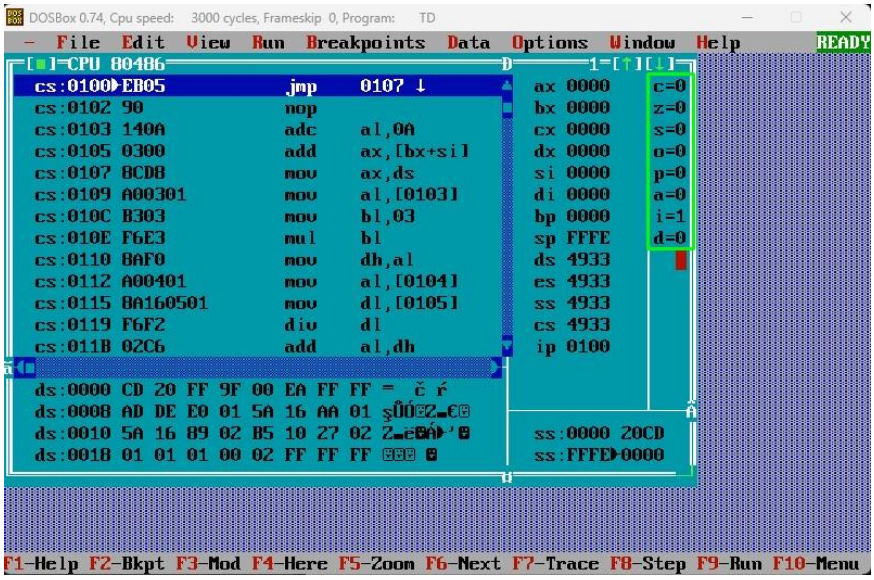
Nie będę ich tłumaczył ale tu jest moim zdaniem spoko link
https://www.eng.auburn.edu/~sylee/ee2220/8086_instruction_set.html

Flagi

Na screenie flagi są oznaczone zielonym prostokątem.

Rejestr statusu (rejestr flag)(nie wiem jak to się nazywa bo znajduje różne nazwy)(nie ma to znaczenia to są po prostu flagi)(kurwa ile nawiasów)(a to jeszcze jeden tak na zaś) ma 16 bitów, każda flaga ma „przypisany” jeden bit.

Nie chce mi się tłumaczyć więc daje tak:



- **Carry Flag (CF)** - this flag is set to **1** when there is an **unsigned overflow**. For example when you add bytes **255 + 1** (result is not in range 0...255). When there is no overflow this flag is set to **0**.
- **Zero Flag (ZF)** - set to **1** when result is **zero**. For none zero result this flag is set to **0**.
- **Sign Flag (SF)** - set to **1** when result is **negative**. When result is **positive** it is set to **0**. Actually this flag take the value of the most significant bit.
- **Overflow Flag (OF)** - set to **1** when there is a **signed overflow**. For example, when you add bytes **100 + 50** (result is not in range -128...127).
- **Parity Flag (PF)** - this flag is set to **1** when there is even number of one bits in result, and to **0** when there is odd number of one bits. Even if result is a word only 8 low bits are analyzed!
- **Auxiliary Flag (AF)** - set to **1** when there is an **unsigned overflow** for low nibble (4 bits).
- **Interrupt enable Flag (IF)** - when this flag is set to **1** CPU reacts to interrupts from external devices.
- **Direction Flag (DF)** - this flag is used by some instructions to process data chains, when this flag is set to **0** - the processing is done forward, when this flag is set to **1** the processing is done backward.

Odpowiedzi do zadania 1

1. Co to znaczy, że kod jest przemieszczalny (relokowalny)?

Jest to kod programu komputerowego, który może być uruchamiany w różnych obszarach pamięci komputera bez konieczności zmiany jego zawartości.

2. Co znajduje się w pliku typu .OBJ?

Plik ma w sobie przemieszczalny moduł pośredni. Kod zawarty w pliku .OBJ (tzw. kod półskompilowany) zawiera już instrukcje programu zakodowane w języku maszyny, ale nie jest jeszcze całkowicie przygotowany do wykonania przez procesor.

3. Jaka jest postać pliku typu .EXE?

Plik typu .EXE jest złożony z sekwencji instrukcji procesora, danych i metadanych programu. Te instrukcje i dane są zapisywane w formacie binarnym zgodnym z architekturą procesora, na który jest przeznaczony program. **Jest relokowalny!**

4. Jaka jest postać pliku typu .COM?

Plik typu .COM jest niczym innym jak tylko zwyczajną mapą pamięci, nie podlegającą w trakcie ładowania żadnym modyfikacjom i uzupełnieniom (w odróżnieniu od .EXE). Musi być napisany tak, by jego pierwszy rozkaz miał przesunięcie 256(100h) – pierwsze 256 bajtów w segmencie zajmie obszar PSP (spełnienie tego gwarantuje nam dyrektywa ORG) Ma maksymalnie 64KB rozmiaru pamięci, czyli jeden segment. Końcowe 256 bajtów jest rezerwowane na stos programowy. Nie jest relokowalny!

5. ~~Kiedy możliwa jest transformacja pliku .EXE na .COM?~~
6. ~~Co zawiera plik typu .LST?~~
7. ~~Co zawiera plik typu .CRF?~~
8. Do czego służy dyrektywa asemblera ORG?

Dyrektywa asemblera ORG służy do określenia adresu początkowego generowanego kodu maszynowego w pamięci operacyjnej. W przypadku programów typu .COM, ORG 100h informuje, że program będzie załadowany do pamięci operacyjnej na adres 100h (256 w systemie dziesiętnym). Robi się tak ponieważ w pierwszych 256 bajtach są przechowane dane dotyczące programu, taki nagłówek nazywamy PSP.

9. Na czym polega proces łączenia modułów?

Proces łączenia modułów polega na połączeniu (konsolidacji) poszczególnych modułów źródłowych (często o rozszerzeniu .obj) w jeden plik wykonywalny. W trakcie tego procesu mogą być również łączone moduły biblioteczne, jeśli są one wykorzystywane w programie.

Konsolidacja odbywa się za pomocą programu zwanego konsolidatorem (linker). Jednym z linkerów jest program „Turbo Linker” (polecenie TLINK).

Konsolidator TLINK pozwala na połączenie wielu modułów w jeden plik wykonywalny (.exe lub .com).

10. Wymień sposoby zakańczania programu ze wskazaniem ich wad i zalet.

Nie do końca wiem o co tu chodzi ale jakieś są.

11. Jak odnaleźć adres PSP w różnych fazach działania programu?

Adres PSP (Program Segment Prefix) w systemie DOS można odnaleźć poprzez przekazanie go jako argumentu podczas uruchamiania programu lub korzystając z rejestru DS w trakcie działania programu. Offsety w instrukcjach asemblera mogą również wskazywać na konkretne dane w obszarze PSP, takie jak parametry linii poleceń

12. Co to jest PSP?

PSP (Program Segment Prefix) jest nagłówkiem segmentu programu tworzonym przez system operacyjny podczas ładowania programu. Zajmuje on pierwsze 256 bajtów w pamięci operacyjnej, do której ładowany jest program. PSP zawiera różne informacje, w tym parametry przekazane w linii poleceń podczas uruchamiania programu, adresy i parametry systemowe. Jest to istotny element dla programów działających pod kontrolą systemu DOS. W PSP przechowywane są także inne ważne informacje, takie jak numer wersji systemu operacyjnego, flagi, rozmiar segmentu stosu, adresy procedur obsługi przerwań itp. Dzięki PSP program może uzyskać dostęp do różnych parametrów i funkcji systemowych, co jest niezbędne do poprawnego działania w środowisku DOS.

13. Podaj przykłady wykorzystania informacji przechowywanej w obszarze PSP.

Parametry z linii poleceń: Programy mogą odczytywać parametry przekazane podczas uruchamiania, takie jak nazwa pliku czy opcje konfiguracyjne. Numer wersji systemu: Programy mogą sprawdzać numer wersji systemu, dostosowując działanie do konkretnej wersji DOS. Adresy procedur przerwań: PSP zawiera adresy procedur obsługi przerwań, co umożliwia korzystanie z różnych funkcji systemowych. Rozmiar segmentu stosu: Informacje o rozmiarze segmentu stosu są istotne dla zarządzania pamięcią w trakcie wykonywania programu. Flagi i ustawienia: PSP może zawierać flagi i ustawienia specjalne, które kontrolują zachowanie programu lub dostosowują je do warunków środowiska.