Komparatory i skoki warunkowe (bezwarunkowe w sumie też)

Generalnie wszyscy już w tym momencie powinni wiedzieć, że procesor wykonuje rozkazy kolejno (sekwencyjnie) od tych na najmłodszym adresie do tych najstarszym. Tłumacząc na polski: program wykonuje się od góry do dołu. Możemy tę kolejność jednak trochę pozmieniać: do tego posłużą nam skoki – warunkowe i bezwarunkowe.

Gdy procesor napotka instrukcję skoku, to przeskoczy do zadanego adresu (może skoczyć do przodu, ale może też do tyłu) i od miejsca lądowania będzie dalej wykonywał instrukcje kroczek po kroczku. W związku z tym trzeba przypilnować, żebyśmy przypadkiem nie stworzyli programu, który nigdy się nie skończy.

Podstawowym i zarazem najprostszym skokiem jest skok bezwarunkowy. W asm będzie do jego wykonania służyła instrukcja

```
jmp etykieta
```

Taki programik, gdyby nie skok wydrukowałby nam literki AB i się zakończył, ale przez to, że po wywołaniu przerwania drukującego literkę A dokonaliśmy skoku do etykiety koniec, to literka B nie została wydrukowana. Proste.

```
org 100h


mov AH, 02h
mov DL, 41h
int 21h

jmp koniec

mov DL, 42h
int 21h

koniec:

mov AX, 4C00h
int 21h
```

Natomiast w insighcie możemy zaobserwować, że jeszcze przed wykonaniem czegokolwiek program wie, do jakiej konkretnie etykiety będzie skakał.

```
0100 B402      mov   ah,02
0102 B241      mov   dl,41
0104 CD21      int   21
0106 E90400    jmp   010D ↓
0109 B242      mov   dl,42
010B CD21      int   21
010D B8004C    mov   ax,4C00
0110 CD21      int   21
```

Uważać jednak trzeba, aby nie przesadzić i nie doprowadzić do sytuacji, w której będziemy wykonywać skok bezwarunkowy do tyłu bez możliwości innego zakończenia programu. Mechanizm skoków bezwarunkowych jest bardzo przydany, ale istnieje duża szansa, że raczej używać będziemy skoków warunkowych, a bezwarunkowego do wychodzenia na koniec programu z ominięciem różnych rzeczy, które moglibyśmy napotkać po drodze.

Skoki warunkowe

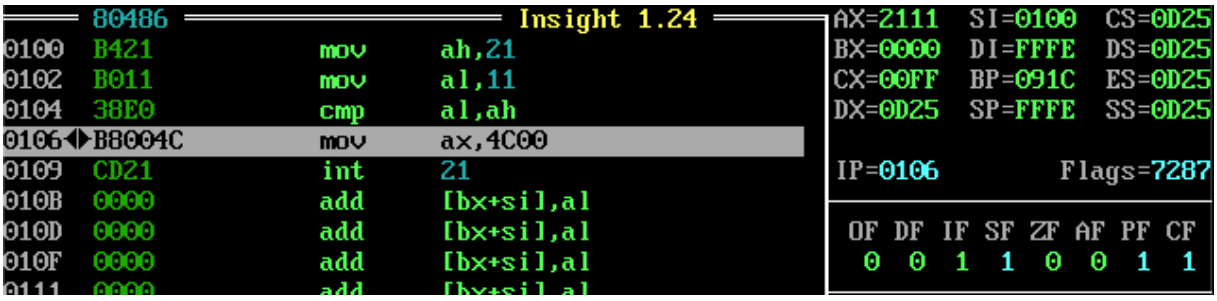
Mechanizm jest dość prosty, tak mi się wydaje w każdym razie. Kiedy procesor napotka instrukcję skoku warunkowego, to skok do etykiety zostaje wykonany tylko w wypadku, w którym warunek jest spełniony. Instrukcji skoków warunkowych mamy całkiem sporo, po wszystkie zapraszam tutaj: https://pl.wikibooks.org/wiki/Asembler_x86/Instrukcje/Skokowe#Warunek, a tu te najczęściej używane:

- JE – skok, gdy A równe B
- JNE – skok, gdy A nierówne B
- JG – skok, gdy A większe od B
- JGE – skok, gdy A większe lub równe B
- JA – skok, gdy A większe od B dla liczb bez znaku
- JB – skok, gdy A mniejsze od B dla liczb bez znaku
- JG – skok, gdy A większe od B (ze znakiem)
- JL – skok, gdy A mniejsze od B (ze znakiem)

Wiemy już bez wątpienia, jak gdzieś skoczyć, gdy coś jest większe albo mniejsze, pozostaje w takim razie ustalić jeszcze czym to „coś” jest. Służyć nam do tego będzie instrukcja cmp.

```
cmp A, B ; w miejscu A i B może być wstawiony rejestr albo liczba, albo cokolwiek innego.
```

W poniższym przykładzie widzimy, że w momencie wykonania instrukcji porównania AL z AH ustawione zostały flagi SF, PF i CF.



Te flagi ustawiane są różnie, w zależności od wyniku porównania. Gdybyśmy w następnym kroku mieli instrukcję skoku, to sprawdziłaby ona status odpowiednich dla niej flag i jeśli byłyby one ustawione, to skok zostałby wykonany, jeśli nie, to program szedłby dalej.

```
org 100h

mov AH, 21h
mov AL, 11h
cmp AL, AH

JL mniejsze
JG koniec

mniejsze:
mov AH, 02h
mov DL, '<'
int 21h
jmp koniec

mov AH, 02h
mov DL, 'E'
int 21h

koniec:
mov AX, 4c00h
int 21h
```

W tym programiku mamy już skok. Porównujemy w nim AL (11h) i AH (21h). Jeśli 11h jest mniejsze od 21h, a jak ostatnio sprawdzałem to tak było, to ustawione przy operacji porównania zostaną flagi SF, FP i CF, IF jest już ustawiona wcześniej, pozostałe są 0. Skok JL wykonuje się, kiedy SF jest różne od OF. W tym wypadku tak jest, bo SF ustawiło się przy porównaniu, a OF pozostało zerem.

Warto przy okazji zwrócić uwagę na to, że tę mini-funkcyjkę „mniejsze” zakończyliśmy bezwarunkowym skokiem do zakończenia programu. Taki zabieg pozwoli na ominięcie kodu, którego nie chcielibyśmy, aby się wykonywał, kiedy gdzieś skaczemy, np. etykiet obsługujące pozostałe wyniki porównania.

To było po inżyniersku, a teraz po ludzku:

Zaczynamy od porównania ze sobą dwóch danych: A i B. Następnie powinny pojawić się skoki do etykiet, zależnie od otrzymanego wyniku porównania. Każda taka etykieta powinna wykonywać jaką akcję i wykonywać bezwarunkowy skok za definicje wszystkich etykiet. Dla ułatwienia można to sobie rozrysować na jakimś diagramie blokowym, czy coś.

Zadanie 1

Napisz programik, który przyjmie od użytkownika jeden znak. Jeśli będzie większy niż ‘_’ (5Fh) to wydrukuje ‘>’, jeśli mniejszy to wydrukuje ‘<’, jeśli równy, to wydrukuje ‘==’

Zadanie 2

Zmodyfikuj programik tak, aby rozpoznawał wielkie i małe litery. Wielkie litery mają kody ASCII 41h-5Ah, a małe 61h-7Ah. Program powinien drukować informacje zwrotną, np. „wielka litera”, „mala litera”, „error”

Zadanie 3 - trudniejsze [[OSTROŻNIE]]

Napisz programik, który w pętli będzie przyjmował i od razu wypisywał przyjęte znaki, ale jeśli wprowadzony zostanie znak Q to się zakończy.