Komendy i kody liczbowe

Instrukcje maszynowe dzielą się na trzy kategorie:
-ruch danych,
-arytmetyczne/logiczne,
-przebieg sterowania.
Przedstawione tutaj są jedynie najważniejsze instrukcje z poszczególnych kategorii.

Ruch danych (Data movement):

- **mov** - instrukcja kopiuje element danych z lokalizacji pierwszego operanda do drugiego.
- **push** - instrukcja umieszcza operand na górze stosu pamięci.
- **pop** - instrukcja usuwa element danych z góry stosu z powrotem do operanda.
- **lea** - instrukcja umieszcza adres określony przez pierwszy operand do rejestru określonego przez drugi operand.

Logiczne i arytmetyczne:

- **add-** dodanie do siebie dwóch operandów, przechowując wynik w drugim operandzie.
- **sub** - odejmowanie dwóch operandów, gdzie operand pierwszy jest odejmowany od drugiego.
- **inc**, **dec** - inkrementacja oraz dekrementacja. Instrukcja zwiększa (inc) lub zmniejsza(dec) zawartość operandu o 1.
- **imul** - instrukcja mnożenia. Ma dwa typy składni- z dwoma lub trzema operandami. W pierwszym typie mnoży przez siebie operandy i wynik przechowuje w tym drugim.
  W drugim mnoży drugi operand z trzecim, a wynik przechowuje w ostatnim. Operand przechowujący wynik musi być rejestrem.
- **idiv** - instrukcja dzielenia, dzieli zawartość 64 bitowego integera EDX:EAX (gdzie EDX jest najbardziej znaczący, a EAX najmniej znaczący) przez zawartość EBX. Iloraz przechowywany jest w EAX, a reszta w EDX.
- **and, or, xor** - logiczne i, lub, żadne z
- **not** - logiczne przeciwieństwo, odwraca wartości wszystkich bitów w operandzie
- **neg** - negacja zawartości operanda
- **shl, shr** - instrukcja przesuwająca bity w operandzie w lewo lub prawo. Puste miejsca uzupełnia zerami.

Przebieg sterowania(Control flow)

- **cmp** - porównaj wartości dwóch określonych operandów
- **call** - Instrukcja call kładzie aktualną lokalizację wykonywanego kodu na stos pamięci, a następnie wykonuje skok do lokalizacji określonej przez etykietę operandu w celu wykonania podprogramu. W przeciwieństwie do instrukcji jmp, instrukcja ta zapisuje lokalizację, aby móc wrócić po wykonaniu podprogramu.
- **ret** - instrukcja implementuje mechanizm powrotu z podprogramu. Wyciąga lokalizację wykonywanego kodu ze stosu pamięci, a następnie powraca do poprzedniej lokalizacji wykonywanego kodu.
- **jmp** - instrukcja przenosi sterowanie do instrukcji w pamięci zlokalizowanej w miejscu określonym przez operand

- **j[..]** - przeniesienie instrukcji w określone miejsce w zależności od zaistniałej sytuacji:
  je <label> (jump when equal) - skocz gdy równy
  jne <label> (jump when not equal) - skocz gdy nie jest równy
  jz <label> (jump when last result was zero) - skocz gdy poprzedni wynik wynosił 0
  jg <label> (jump when greater than) - skocz gdy większy niż
  jge <label> (jump when greater than or equal to) - skocz gdy większy lub równy
  jl <label> (jump when less than) - skocz gdy mniejszy
  jle <label> (jump when less than or equal to) - skocz gdy mniejszy lub równy