Регистры общего назначения:

целые вещественные

(2 байта) (4 байта)

ah eax

bh ebx

ch ecx

dh edx

eh eex

sp - указатель на вершину стека целых чисел

esp - указатель на вершину стека вещественных чисел

pc - указатель на текущую команду

Стэк растёт вверх. По умолчанию целый и вещественный стэк

инициализируется адресом, который идёт сразу после кода.

Команды:

mov 1 ah

1 - что кладём

eax - куда кладём

1, 10, 23 - обычные числа

1.0, 5.3, 4.2 - дробные числа

ah, bh, ch - обычные регистры

eax, efx - вещественные регистры

%1, %10, %3 - обращение к значениям по абсолютным адресам

%ah, %bh, %ch - обращение по адресу, записанному в регистре (только для целых регистров)

add 4 eh

аргументы - простые числа, вещественные числа, простые регистры, вещественные регистры

4 - на сколько увеличиваем (числа или регистры из которых брать числа)

eh - что увеличиваем (регистры)

sub 6 dh

аргументы - простые числа, вещественные числа, простые регистры, вещественные регистры

То же самое, только уменьшаем

mul ch

аргумент - простое число, вещественное число, простой регистр, вещественный регистр

умножает регистр ah (или eax) на аргумент (результат в регистр ah ил eax)

аргумент может быть числом или регистром

div fh

аргумент - простое число, вещественное число, простой регистр, вещественный регистр

делит регистр ah (или eax) на аргумент. Остаток от деления в eh

аргумент может быть числом или регистром

label: - данная команда описана для условных/безусловных переходов. Т.е.

чтобы нам не писать адрес в О/П, на который будет перепрыгивать программа, мы будем указ.

такие метки (а транслятор соотв. будет подст. все необх. адреса).

метка, может иметь любое имя

чтобы получить адрес метки, пишем label

чтобы получить значение, по адресу метки, пишем %label

//Замечание!

//Сложность реализации команды – label, выражается в том, что в интерпретаторе придется //выполнять пару проходов:  
//П1 – Вычисление адресов меток;

//П2 – Выставление адресов, как «конкретных»/«вшитых».

jmp 31

jmp label

переход выполнения на метку, т.е. перепрыгивание команды на указанный адрес (безусловный переход).

аргумент - адрес (целое число), метка, или регистр в котором лежит адрес.

call 31

call label

размещение на стэке адреса возврата, переход выполнения на метку,

аргумент - адрес (целое число), метка, или регистр в котором лежит адрес

ret

команда без аргументов

Извлекает из стэка адрес, прыгает по этому адресу

cmp ah 10

сравнение двух аргументов

аргументы - простые числа, вещественные числа, простые регистры, вещественные регистры

если равно, заносит 0 в eh

если первый аргумент больше заносит 1 в eh

если второй аргумент больше заносит -1 в eh

//Следующие 3 конструкции – это аналог конструкции – “if”, в я/п

//высшего уровня.

je label

прыжок если в eh 0 (числа равны)

jg label

прыжок если в eh 1 (первый аргумент больше)

jl label

прыжок если в eh -1 (первый аргумент меньше)

push ch

аргумент - простое число, вещественное число, простой регистр, вещественный регистр

положить значение из ch на стэк

pop bh

аргумент - простое число, вещественное число, простой регистр, вещественный регистр

извлечь значение со стэка в bh

pass

пустая команда без аргументов

int 10

прерывания – способ взаимодействия с внешней системой. К примеру, если необходимо, чтобы

какая-то «железка» вывело сообщение на экран, вызываем одно прерывание…

Прерывания будет для:

- Текстовой информации;  
- Завершения программы;

- Эмуляция графической системы

… (см. ниже)

аргумент - цел\_число

Данные

.byte 48 - однин байт

.short 8912 - целое число, два байта

.float 23.5 - дробное число, четыре байта

.string Hello, world! - автоматически вставляется строку

(конвертирует в ascii, добавляет 0-й символ)

.ascii Hello, world! - тоже самое, только без добавление 0-го символа

.space 10 - пропускает заданное количество байт (ничего не кладёт)

Команды занимают: 1 байт

Номера (названия) регистров: 1 байт

Целые числа занимают: 2 байта

Дробные числа: dh 4 байта

Список прерываний:

10 - выход из программы

11 - вывод числа из регистра ah на экран

12 - вывод числа из регистра eax на экран

13 - ввод числа в регистр ah (с клавиатуры)

14 - ввод числа в регистр eax (с клавиатуры)

15 - вывод текста на экран, лежащего по адресу из регистра ah до 0-го символа

16 - ввод текста с клавиатуры в память по адресу из ah. В bh будет лежать длина введенного текста

17 - сброс в файл отладочной информации (дамп регистров и памяти)

18 - вывод текста на экран, лежащего по адресу из регистра ah до 0-го символа, с переводом строки

Обозначения регистров:

Регистры обозначаются номером (соотв. при генерации оп/машинного-кода), занимают 1 байт:

0 - ah

1 - bh

2 - ch

3 - dh

4 - eh

5 - sp

6 - pc

7 - rh

0 - eax

1 - ebx

2 - ecx

3 - edx

4 - eex

5 – esp

//Замечание!

//Для установления, используемой команды (в случае если совпадают оп-коды), программа //(виртуальная машина) будет ориентироваться по контексту.

Байт-коды команд:

//Замечание!

//Байт-коды ниже, обозначены с пом. чисел, перед описанием самой команды.

OP\_MOV\_1 1 mov цел\_число цел\_регистр

OP\_MOV\_2 2 mov цел\_число память

OP\_MOV\_3 3 mov цел\_число память\_из\_регистра

OP\_MOV\_4 4 mov дроб\_число дроб\_регистр

OP\_MOV\_5 5 mov дроб\_число память

OP\_MOV\_6 6 mov дроб\_число память\_из\_регистра

OP\_MOV\_7 7 mov цел\_регистр цел\_регистр

OP\_MOV\_8 8 mov цел\_регистр память

OP\_MOV\_9 9 mov цел\_регистр память\_из\_регистра

OP\_MOV\_10 10 mov дроб\_регистр дроб\_регистр

OP\_MOV\_11 11 mov дроб\_регистр память

OP\_MOV\_12 12 mov дроб\_регистр память\_из\_регистра

OP\_MOV\_13 13 mov память цел\_регистр

OP\_MOV\_14 14 mov память дроб\_регистр

OP\_MOV\_15 15 mov память\_из\_регистра цел\_регистр

OP\_MOV\_16 16 mov память\_из\_регистра дроб\_регистр

OP\_MOV\_17 17 mov цел\_регистр дроб\_регистр

OP\_MOV\_18 18 mov дроб\_регистр цел\_регистр

OP\_PASS 19 pass (пустая команда)

OP\_INT 20 int (прерывание, аргумент - цел\_число)

OP\_ADD\_1 22 add цел\_число цел\_регистр

OP\_ADD\_2 23 add цел\_регистр цел\_регистр

OP\_ADD\_3 24 add дроб\_число дроб\_регистр

OP\_ADD\_4 25 add дроб\_регистр дроб\_регистр

OP\_SUB\_1 26 sub цел\_число цел\_регистр

OP\_SUB\_2 27 sub цел\_регистр цел\_регистр

OP\_SUB\_3 28 sub дроб\_число дроб\_регистр

OP\_SUB\_4 29 sub дроб\_регистр дроб\_регистр

OP\_MUL\_1 30 mul цел\_число

OP\_MUL\_2 31 mul цел\_регистр

OP\_MUL\_3 32 mul дроб\_число

OP\_MUL\_4 33 mul дроб\_регистр

OP\_DIV\_1 34 div цел\_число

OP\_DIV\_2 35 div цел\_регистр

OP\_DIV\_3 36 div дроб\_число

OP\_DIV\_4 37 div дроб\_регистр

OP\_JMP\_1 40 jmp число

OP\_JMP\_2 41 jmp цел\_число

OP\_JE\_1 42 je число

OP\_JE\_2 43 je цел\_число

OP\_JG\_1 44 jg число

OP\_JG\_2 45 jg цел\_число

OP\_JL\_1 46 jl число

OP\_JL\_2 47 jl цел\_число

OP\_JNE\_1 48 jne число

OP\_JNE\_2 49 jne цел\_число

OP\_CMP\_1 50 cmp цел\_число цел\_регистр

OP\_CMP\_2 51 cmp цел\_регистр цел\_регистр

OP\_CMP\_3 52 cmp дроб\_число дроб\_регистр

OP\_CMP\_4 53 cmp дроб\_регистр дроб\_регистр

OP\_PUSH\_1 60 push цел\_число

OP\_PUSH\_2 61 push цел\_регистр

OP\_PUSH\_3 62 push вещ\_число

OP\_PUSH\_4 63 push вещ\_регистр

OP\_POP\_1 65 pop цел\_регистр

OP\_POP\_2 66 pop вещ\_регистр

OP\_CALL\_1 70 call память

OP\_CALL\_2 71 call память\_из\_регистра

OP\_RET 75 ret

OP\_AND\_1 76 and цел\_число цел\_регистр

OP\_AND\_2 77 and цел\_регистр цел\_регистр

OP\_OR\_1 78 or цел\_число цел\_регистр

OP\_OR\_2 79 or цел\_регистр цел\_регистр

OP\_XOR\_1 80 xor цел\_число цел\_регистр

OP\_XOR\_2 81 xor цел\_регистр цел\_регистр

OP\_NOT 82 not цел\_регистр