Commands

|  |  |  |
| --- | --- | --- |
| Mnemonic | Parameters | Description |
| mov | <destination>,<source> |  |
| nop |  |  |
| Add | <destination>,<source> |  |
| Sub | <destination>,<source> |  |
| hlt |  |  |
| Crt\_printf | <source> eg. ADDR string |  |
| MessageBox |  |  |
| ExitProcess | Eg. ExitProcess,0 |  |
| Sleep | How long in ms |  |
| version |  |  |
| readInteger |  |  |
| readIntegerWithMessage | Invoke rIWM,ADDR message | Stores value in eax |
| writeInteger | <source> | Stores value in eax |
| neg | <dst> | Negate value |
| adc | <dst>,<src> | Add with carry |
| sbb | <dst>,<src> | Substract with borrow |
| inc | <dst> | Increments signed value by one |
| dec | <dst> | Decrements value by one |
| mul | <src> | Unsigned multiplication of eax by src |
| div | <src> | Unsigned division of eax by src |
| imul | <src>  <dst>,<src>  <dst>,<src1>,<src2> | Signed multiplication |
| idiv | <src> | Signed division |
| And | <dst>,<src> |  |
| Or | <dst>,<src> |  |
| Xor | <dst>,<src> |  |
| not | <dst> | Change 0 to 1 and 1 to 0 |
| Shr | <dst>,<count> | Shift logical right |
| Shl | <dst>,<count> | Shift logical left |
| sar | <dst>,<count> | Shift arithmetic right |
| **LED SUBROUTINES** | | |
| Sleep | <How long to wait in ms> |  |
| writeRow | <row>,<bit value> | Writes row |
| readRow | <row> | Reads in supplied row to eax |
| setPattern | <1-6> | Stes a pattern depending on number given |
| random | <range> | GIVES –range=>+range so negation is necessary if you only need the positive numbers |

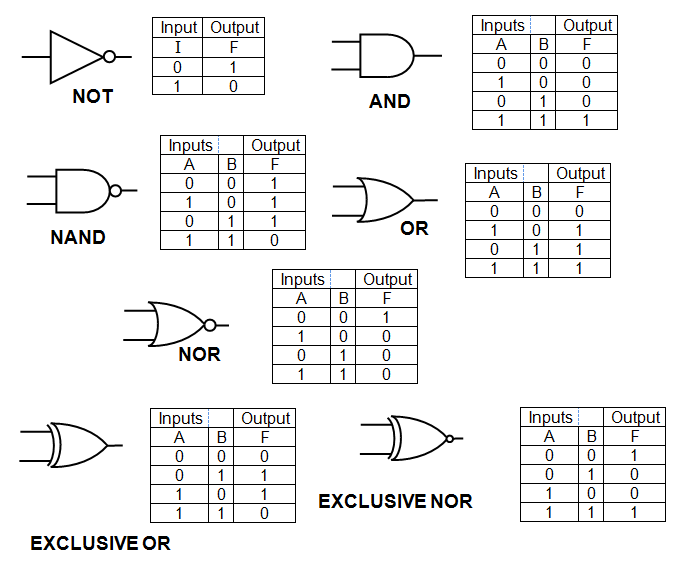
Jump Instructions

|  |  |  |  |
| --- | --- | --- | --- |
| Mnemonic | Description | Signed-ness | Flags |
| cmp | <dest>,<src> Subtracts src from dest but doesn’t store the result |  |  |
| jmp | Always |  |  |
| Jne | != |  | ZF=0 |
| Je | = | unsigned | ZF=1 |
| Ja | > | unsigned | CF=0 ZF=0 |
| Jae | >= | unsigned | CF=0 ZF=1 |
| Jb | < | unsigned | CF=1 |
| Jbe | <= | unsigned | CF=1 ZF=1 |
| Jg | > | signed | ZF=0 SF=of |
| Jge | >= | signed | SF=of |
| Jl | < | signed | SF != of |
| jle | <= | signed | ZF=1 or SF!= of |

Notes

* Multiplication works with 64 bits no matter what, when you multiply if it goes over 32 bits the rest of the value goes into edx
* Division gives you the quotient (answer) and the remained. The quotient goes into eax while the remainder goes into edx
* When shifting left or right the last bit will be moved into the carry bit, using this we can shift right and check what the last number was (0/1) by incrementing at the occurrence of the desired number
* Using AND/OR will allow us to see what bits have changed

* Remember the sizes for each variable
  + byte "db" = 1 byte
  + word or “dw” = 2 bytes
  + dword or “dd” = 4 bytes
* using sbyte,sword and sdword can be used for integers
  + don’t have to it just makes differentiating easier
* string BYTE ‘this is an ascii character %c’,0
  + crt\_printf, ADDR string, num(between 32 and 127)
* string BYTE  ‘this is a number %d’,0
* newLine BYTE 13,10,0 (this will print a new line)
* stringWithNewLine BYTE ‘this is a line’,10,0 (this will print a new line at the end of the message)
* Largest 32 bit signed integer: [2147483647](tel:2147483647) (7FFFFFFF)
* Largest 32 bit unsigned integer: 4294967295 (can’t be in hex)
* When writing hex do it like “07FFFFFFFh”



Code Examples

**Skipping negative numbers**

invoke random, 32

cmp eax,0

jge skipNeg

neg eax

skipNeg:

mov randNum, eax