29 x86 Assembly

Wednesday, November 11, 2015 10:02

Linker understand names of things
    Linker looks at all the libraries that I have on my system.
    What if there are multiple puts functions? It will first look for one that you wrote, then look a the libraries on the system
Preprocessor takes all the includes away

Learning process works for any assembly language
Doh.c
```
Int main(void) {
    Return 47;
}
```
-m32, LDFLAGS = -m32 -static
After main functions, returns value to the parent process, in this case the shell
Environment variable in the shell
```
Echo $?
47
Ls
Echo $?
0
```

.global puts main in all registers have % signs in front of them
Sample registers
    %ebp
    %esp
In the 70's when they first made the 8086, about the same as the 6502, basically the 6502 with a different instruction center. Back then the registers were called A, B, C, and were 8-bit. At some point, they decided they decided people want ed bigger registers. So when the 80186 in the 80's, they decided they wanted it to be compatible to the old stuff, made it a 16-bit processor. Added an x at the end, A register an 8-bit register, if you wanted the 16-bit register, access Ax (A extended). In the late 80s, wanted 32-bit registers. Called the new ones EAX (Extended A Extended),

Mov1, moves the value 64 into the register eax.
Constant has a dollar sign in front of it.
Source, destination

AT&T syntax

There are two competing syntatical instrucionts
AT&T - mov is source, destination
INTEK - move is destination, source

Peter is an intel hater, "this is an AMD machine", assignment is in AT&T

syntax

Result of a function goes to eax

```
Int a;
Int b;
Int main(void) {
        Return a + b;
}
```

.comm a means give me a vairable called a

% sign is register
$ is a literal
$0x is a hex literal
A, or w.e. is a address reference