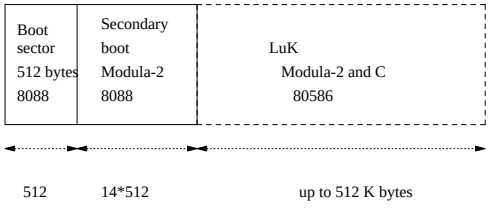
How does the come to life?

- the consists of a collection of Modula-2 modules
- the modules required for laboratory 1, 2 and 3 are different
  - when you compile using `make` the `makefile` is analyzed and a module dependency list is created
  - this list of modules is compiled and linked. An executable file is created (`exe`).
- this executable is an i586 executable and contains data, code and symbol information.
- the i586 needs to be made part of a boot floppy

How does a floppy boot disk work?

- a bootable floppy disk operates in the following way
  - once the PC has been reset the BIOS will attempt check so see whether a floppy disk is in drive A:
  - (providing the BIOS allows it to boot from the floppy disk drive)
- if a floppy disk is present then it reads the first 512 bytes (the first sector) into memory at location 07C00H
  - the BIOS then jumps to location 07C00H

Overview of the boot stages

- three boot phases
  - *primary* boot stage (boot sector, 1 sector, assembly language)
  - *secondary* boot stage (up to 14 sectors 8088 small mode Modula-2)
  - (up to 512K of i586 code, Modula-2)
- 

boot disk

- 512 bytes boot sector is small! Just enough space to place an assembly language program which loads in a larger program
  - loads in *secondary* boot stage at 90200H
  - jumps to 90200H
- *secondary* boot stage
  - consists of limited amounts of assembly language
  - most of the code is written in Modula-2 but compiled to small mode 8088
  - the secondary stage may be up to 14 sectors in size (14 * 512 bytes)

Secondary boot stage

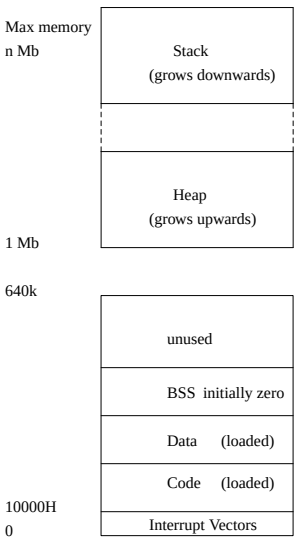
- purpose of *secondary* boot stage is to load in your i586 code as quickly as possible
  - it uses whole track reads whenever possible (fast)
  - the *primary* boot stage only used single sector loads (slow)
  - it loads your Modula-2 i586 executable into location 10000H
  - collects vital statistics about the PC (how much memory the PC contains and where video memory starts)
  - saves this information
  - turns the floppy disk motor off
- finally *secondary* boot stage puts itself into 32 bit mode and calls your i586 code

Boot phase in more detail

- *how* do you put the in the right place?
  - tip, think backwards
- start with the final position you desire
  - and consider how you can achieve it
  - draw memory maps of the different bootstage intemediate positions

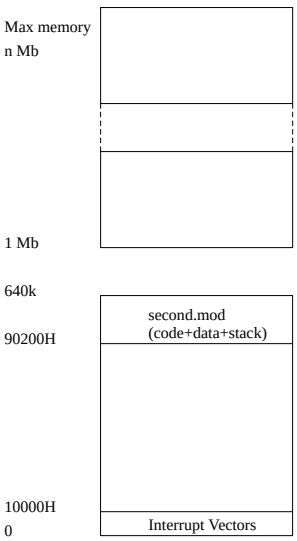
Final memory map for

-



Second memory map for

-



Boot memory map for