TED ÜNİVERSİTESİ

Computer Organization – CMPE361

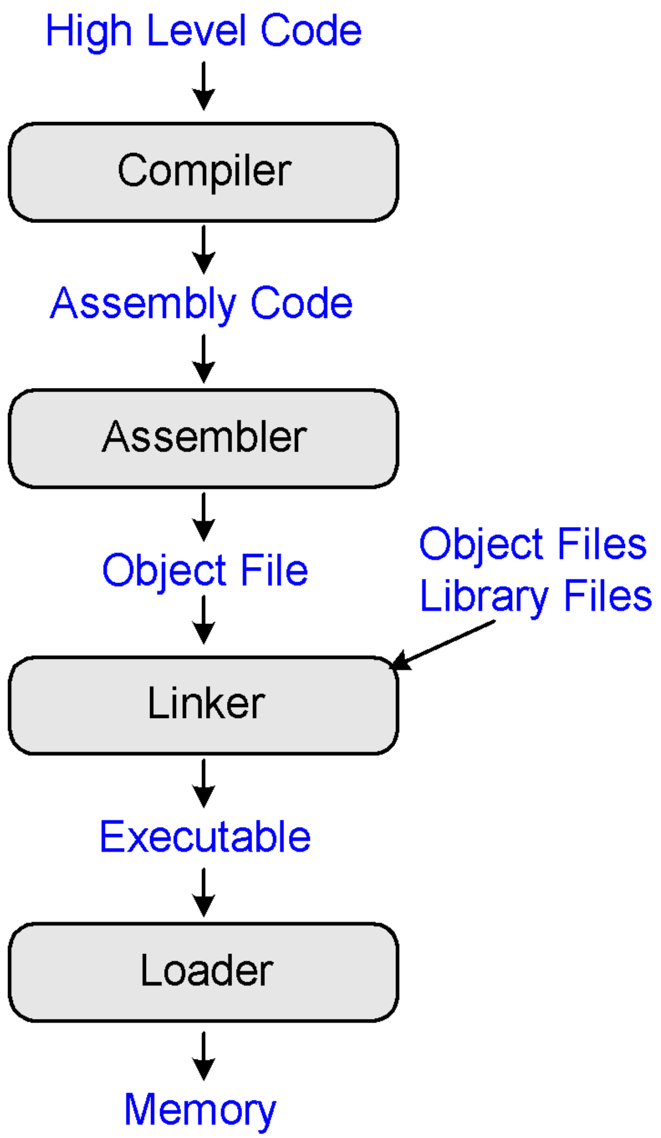
Department of Computer Engineering

TED University- Fall 2023

Compiling, Assembling, Linking, Loading

These Slides are mainly based on slides of the text book (downloadable from the book’s website).

Program: from source to execution

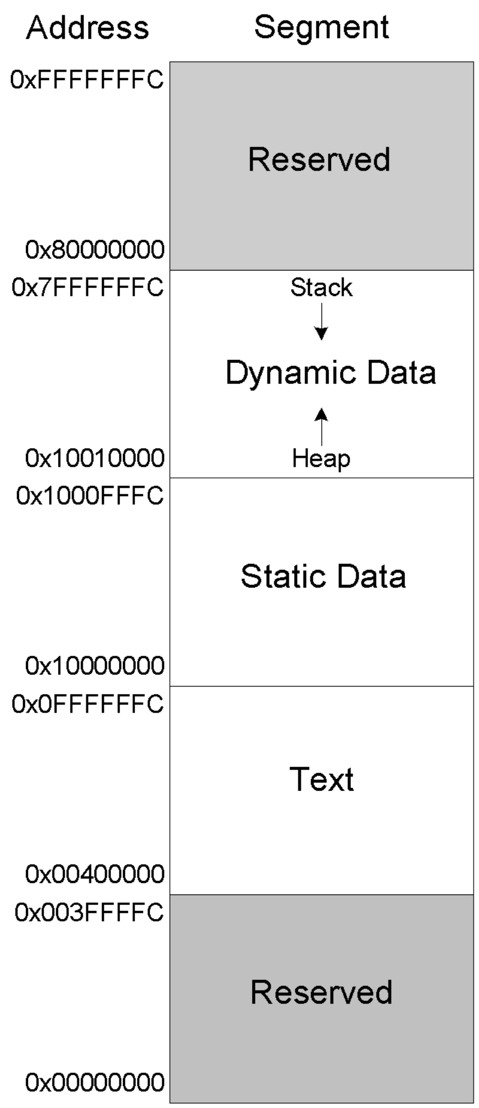


What is Stored in Memory?

- Text: Instructions
- Data:
  - Global or static: allocated before program begins
  - Dynamic: allocated during execution

- How big is memory?
  - At most 2^{32} bytes = 4 GigaBytes (4 GB)
  - Lowest address: 0x00000000
  - Highest address: 0xFFFFFFFF

MIPS Memory Map



Address	Segment
0xFFFFFFFC 0x80000000	Reserved
0x7FFFFFFC 0x10010000	Stack Dynamic Data Heap
0x1000FFFC 0x10000000	Static Data
0x0FFFFFFC 0x00400000	Text
0x003FFFFC 0x00000000	Reserved

Object file: Compiling and Assembling

- In MIPS, a compiler can translate high-level code into assembly or or directly machine code.
- The assembler makes two passes through the assembly code to form machine code.
- **On the first pass**, the assembler assigns instruction addresses, add the labels and global variable names to a table known as **Symbol-Table**.
- The symbol addresses needs two passes.
- Global variables are assigned addresses in the global data segment of memory.
- The creation of machine language code is completed after the second pass through the program.
- The machine code and symbol table are stored as the object file

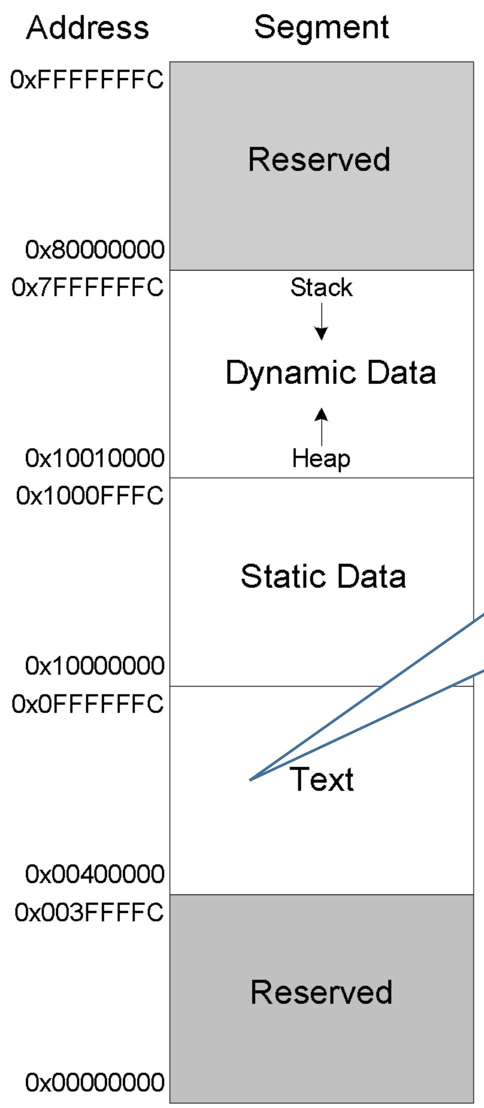
Linker

- The job of the linker is to combine all of the components of a program into a file called the executable.
- The linker **relocates** data and instructions: True addresses are the one in the memory, after linking phase
- The symbol tables is used to adjust the addresses of global variables and labels.

Loading an executable program

- The loader (or OS) loads a program by reading the text segment of the executable file from a storage device (usually the hard disk) into the text segment of memory.
- It sets $gp to 0x10008000 (the middle of the global data segment) and $sp to 0x7FFFFFFC (the top of the dynamic data segment),
- Then performs a jal 0x00400000 to jump to the beginning of the program

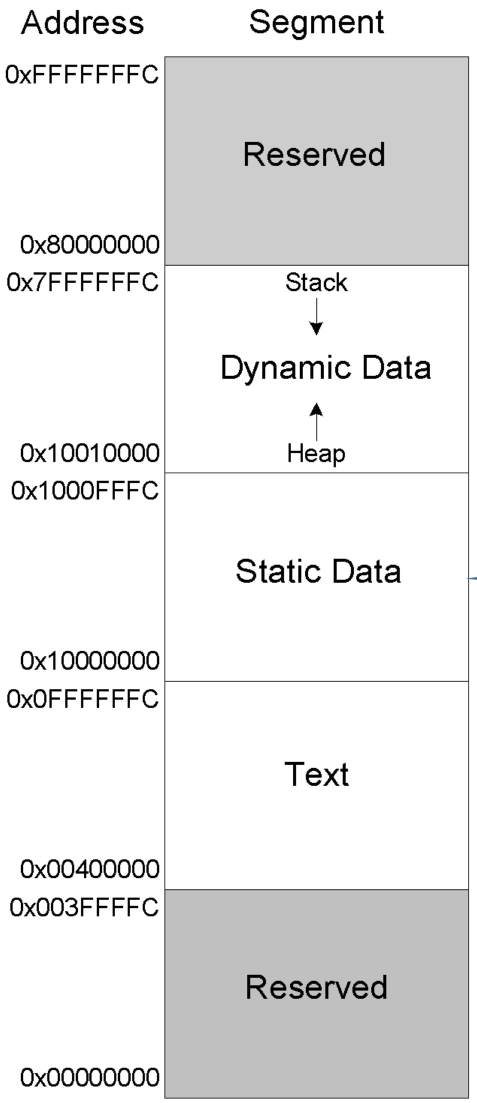
MIPS Memory Map



Address	Segment
0xFFFFFFFC 0x80000000	Reserved
0x7FFFFFFC 0x10010000	Stack Dynamic Data Heap
0x1000FFFC 0x10000000	Static Data
0x0FFFFFFC 0x00400000	Text
0x003FFFFC 0x00000000	Reserved

- Text(program) size= 252 MB
- j instruction can directly jump to any address in the program
- Note that 4 msb of any address are 0…

MIPS Memory Map

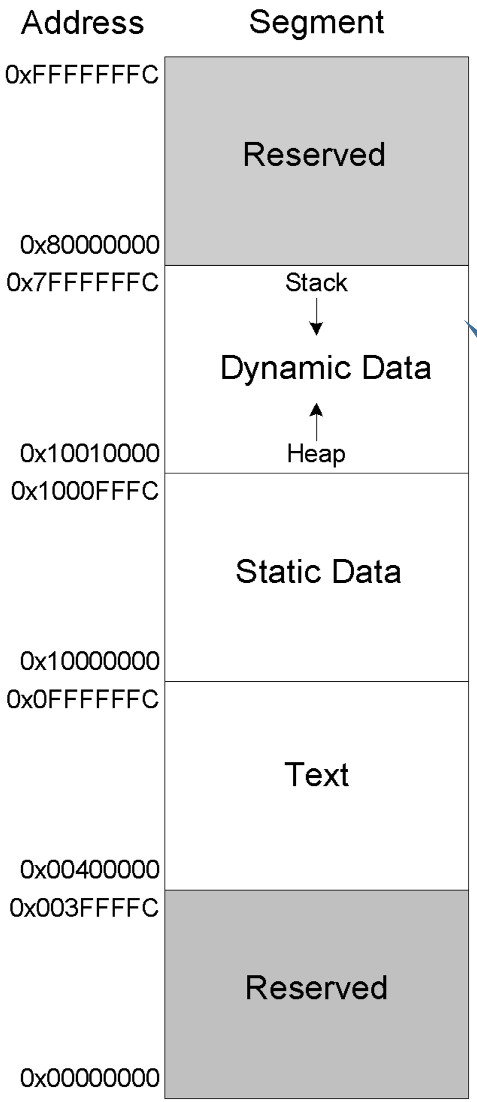


Address	Segment
0xFFFFFFFC 0x80000000	Reserved
0x7FFFFFFC 0x10010000	Stack Dynamic Data Heap
0x1000FFFC 0x10000000	Static Data
0x0FFFFFFC 0x00400000	Text
0x003FFFFC 0x00000000	Reserved

Static data:

- Size=64 KB
- Stores data relative to $gp
- Also note that $gp fixed to 0x10008000

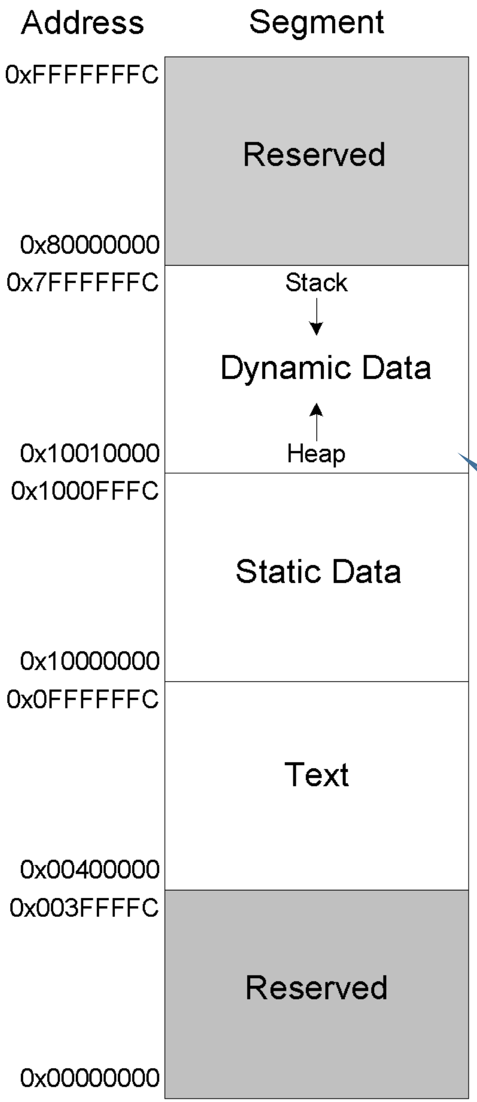
MIPS Memory Map



Stack and heap area

- Stack for local variable
- Stack and heap should never run into eachother

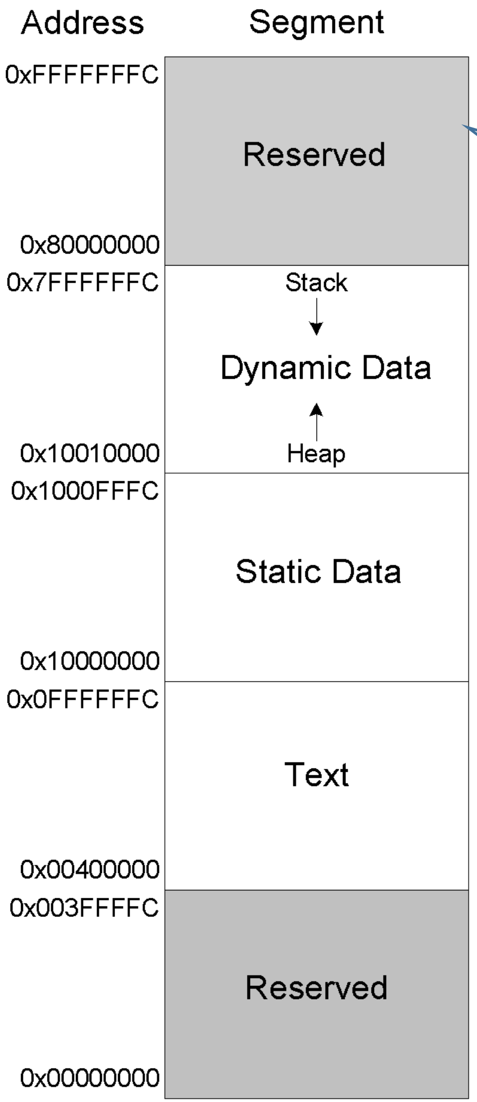
MIPS Memory Map



Heap:

- managed by the programmer

MIPS Memory Map



Reserved memory space:
Used by OS for interrupts and memory mapped I/O