Layout of MIPS program: Code Segment

- begins with:
  .text
  .globl main

main:

• The assembler converts the instructions into 32-bit opcodes.
• The opcodes are loaded onto text/code memory
• Each instruction type (R, I or J type) is converted differently.
• For more info: [http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html](http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html)
Layout of MIPS program: Data Segment

• Begins with
  – .data

• The data put here is loaded as-is into the data memory, with the label retained as reference.

• The size of each data element depends on its type.
Data Segment Examples

• Create strings of text
  
  msg:       .asciiz “hello world”

• Create arrays (integer) of data
  
  A:       .word 0,1,2,3,4,5,6,7,8,9

• Allocate a chunk of memory (in bytes)
  
  empty:   .space 400
**Syscalls**

- MIPS passes the control to the OS to execute this. Usually when we need something interactive to be done.
- Performs a variable service dependent upon the values in various registers.
- $v0 specifies the operation
- $a0 (typically) specifies the argument
  - Other places arguments can come from:
    - $a1, $a2, $a3 – for operations with multiple args
    - $f12 – for floating point number calls
- Sample call (prints 20 on the terminal):
  
  ```
  li $v0, 1
  li $a0, 20
  syscall
  ```
Common Syscalls Examples

• Print integer
  – $v0 = 1, $a0 = integer to print

• Print float
  – $v0 = 2, $f12 = float to print

• Print string
  – $v0 = 4, $a0 = string_address

• Print character
  – $v0 = 11, $a0 = character to print

• Read integer
  – $v0 = 5; after call, $v0 = integer read

• Read character
  – $v0 = 12; after call, $v0 = character read

• Quit execution
  – $v0 = 10

• More info:
  http://students.cse.tamu.edu/wanglei/csce350/reference/syscalls.html