ECECS 314 Section 2

Assembler Directives:

To assist assembler to generate the machine codes

.text	tells assembler that following is part of codes
.data	following is part of data area
.ascii str	insert ascii string into next part of memory
.asciiz str	same as above, ad null byte at end
.word n1,n2	reserve space for words and store values n1,n2 etc
.half n1, n2	reserve space for halfwords and store values n1,n2 etc
.byte n1,n2	reserve space for bytes and store values n1,n2 etc
.space n	reserve space for n bytes
.align m	align the next datum on 2^m byte boundary.

There can be many text and data direstives in one program

pseudo instructions: help programming; do not exist in real machine but can be built from a few others.

move $2, $3 addu $2,$0, $3

li $4, 234	ori $4, $0,234
li $4, big	lui $4,[big-hi]
	ori $4, $4, [big-lo]
la $4, 8($8)	addui $4, $8, 8
la $4, label	lui $1, [label-hi]
	ori $4,$1, [label-lo]

Register name:

$v0, $v1,	$2-$3 function results.
$a0 … $a4	$4 .. $7 arguments
$t0 … $t7	$8 … $15 temporaries
$s0 … $s7	$16 … $23 saved temporary
$t8, $t9	$24, $25 temporary
$k0, $k1	$26, $27 kernel temporaries
$gp	$28 global pointer
$sp	$29 stack pointer
$fp	$30 frame pointer
$ra	$31 return address

Writing an if statement in C code

```
int a;
.....
if (a >= 10 && a <= 15) {
      a = a -10;
      ...
} else {
      printf("Value out of range\n");
      ...
}
```

Writing the same if statement in assembly language.
Here we use the register $t1 to hold the integer value.

```
.data
mesg: .asciiz "Value out of range\n" #stores the string "Value out of
      range"
.text
if:   slti $1, $t1, 10
      bgtz $1, else     #if $t1 is less than 10 goto else
      slti $1, $t1,16
      blez $1, else     #if $t1 is also greater than 15 goto else
      addi $t1, $t1, -10 #otherwise subtract 10 from $t1
      ...
      j exit
else:
      li $v0, 4 # print the message to the screen
      la $a0, msg
      syscall
      ...
exit:
```

An example of a while loop in C:

```
int a = 0;
while (a < 10) {
      a += 2;
      ...
}
```

In assembly code the while loop looks like this:

```
#Assume register $t1 holds the value of variable a.
loop: slti $1, $t1, 10
      blez $1, exit
      addi $t1, $t1, 2
      ...
      j loop
exit:
```

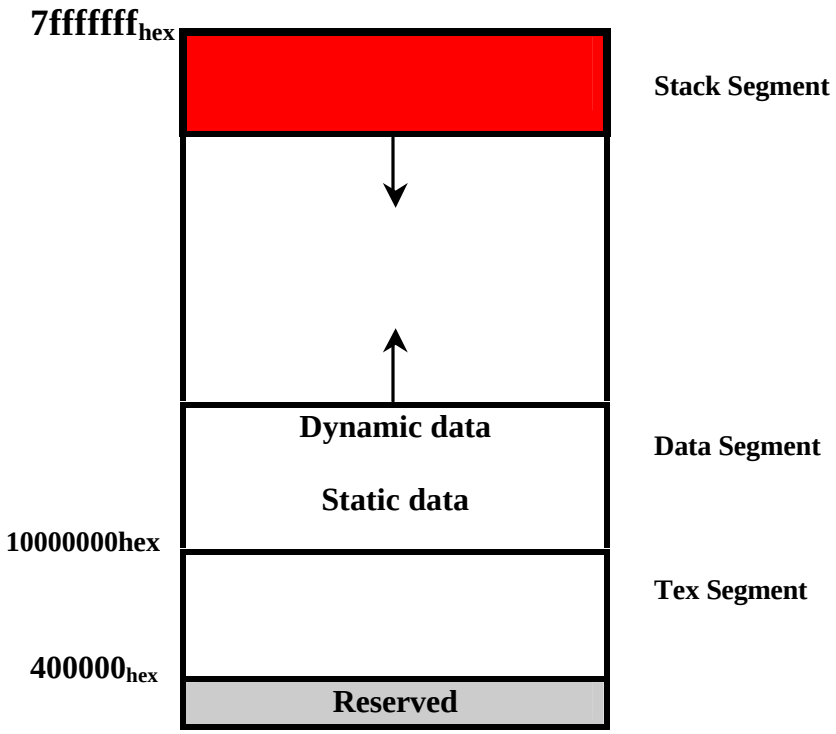
Calling Conventions

Last section: register usage has been covered.

Briefly review it here:

$sp(29): stack pointer
$fp(30): frame pointer
$ra(31): return address

MIPS machine memory Layout



- **Arguments and Return Values**

$a0-$a3(4-7): used to pass the first four arguments to routines
(how to pass the remaining arguments? - use stack!)
$v0,v1(2,3): used to return values from functions

- **Saved Register**

$t0-$t9(8-15,24,25): caller-saved registers – used to hold temporary quantities that need not be preserved across calls