Non-Leaf Procedures

- Procedures that call other procedures
- For nested call, caller needs to save on the stack:
  - Its return address
  - Any arguments and temporaries needed after the call
- Restore from the stack after the call

Non-Leaf Procedure Example

- C code:

```
int fact (int n)
{
  if (n < 1) return f;
  else return n * fact(n - 1);
}
```

  - Argument n in $a0
  - Result in $v0

Non-Leaf Procedure Example

```
int fact (int n)
{
  if (n < 1) return f;
  else return n * fact(n - 1);
}
```

- MIPS code:

```
   fact:
        addi $sp, $sp, -8     # adjust stack for 2 items
        sw   $ra, 4($sp)      # save return address
        sw   $a0, 0($sp)      # save argument
        slti $t0, $a0, 1      # test for n < 1
        beq  $t0, $zero, L1
        addi $v0, $zero, 1    # if so, result is 1
        addi $sp, $sp, 8      #   pop 2 items from stack
        jr   $ra              #   and return
   L1:  addi $a0, $a0, -1     # else decrement n
        jal  fact             # recursive call
88 ⇒    lw   $a0, 0($sp)      # restore original n
        lw   $ra, 4($sp)      #   and return address
        addi $sp, $sp, 8      # pop 2 items from stack
        mul  $v0, $a0, $v0    # multiply to get result
        jr   $ra              # and return
```

addi $a0, $a0, 2

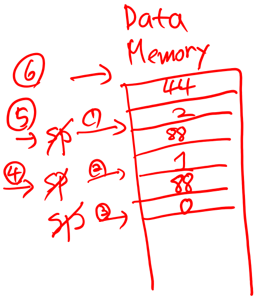
jal fact ⇐ 44

$ra ~~44~~ ~~88~~ ~~88~~ ~~88~~ 44

$a0 ~~2~~ ~~1~~ ~~0~~ ~~1~~ 2

$t0 0

$v0 1*1*2



Character Data

- Byte-encoded character sets
  - ASCII: 128 characters
    - 95 graphic, 33 control
  - Latin-1: 256 characters
    - ASCII, +96 more graphic characters
- Unicode: 32-bit character set
  - Used in Java, C++ wide characters, …
  - Most of the world's alphabets, plus symbols
  - UTF-8, UTF-16: variable-length encodings

Byte/Halfword Operations

char a = 'A';
cout << (int) a;

- Could use bitwise operations
- MIPS byte/halfword load/store
  - String processing is a common case

```
lb rt, offset(rs)      lh rt, offset(rs)
```

Sign extend to 32 bits in rt

```
lbu rt, offset(rs)     lhu rt, offset(rs)
```

Zero extend to 32 bits in rt

```
sb rt, offset(rs)      sh rt, offset(rs)
```

Store just rightmost byte/halfword

CSULB

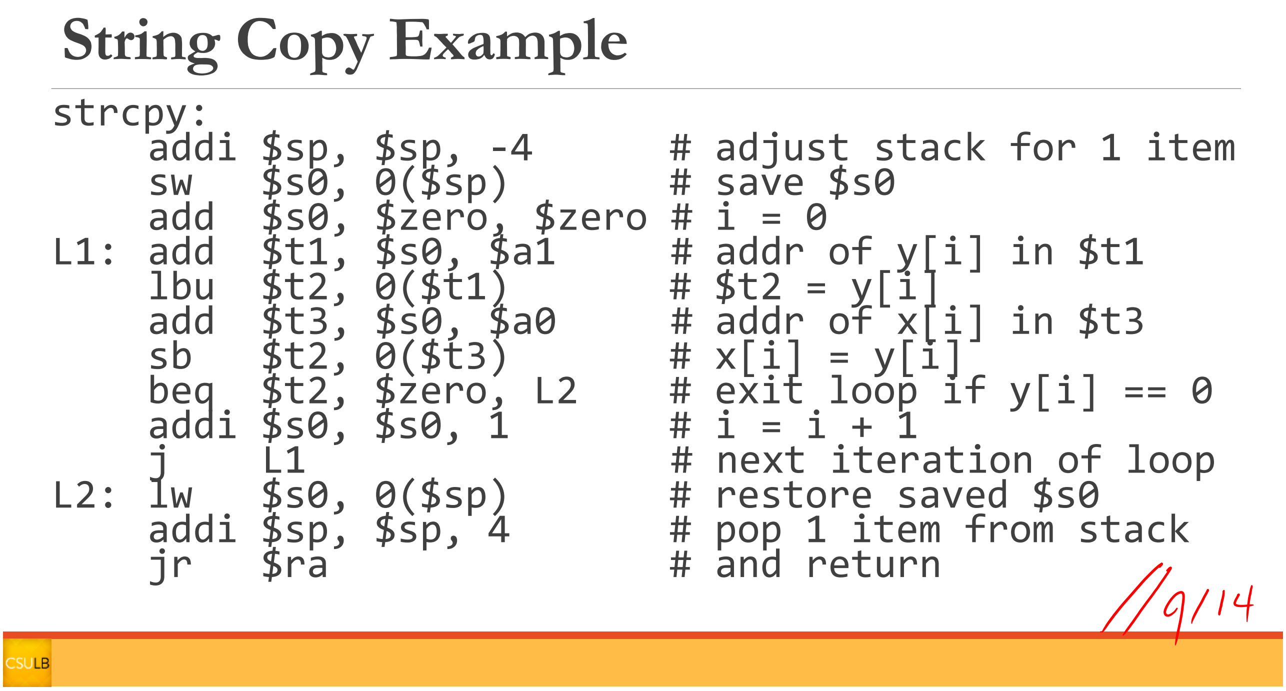
String Copy Example

- C code (naïve):
  - Null-terminated string

```
void strcpy (char x[], char y[]) {
   int i;
   i = 0;
   while ((x[i]=y[i])!='\0')
      i += 1;
}
```

  - Addresses of x, y in $a0, $a1
  - i in $s0

CSULB



String Copy Example

```
strcpy:
    addi $sp, $sp, -4      # adjust stack for 1 item
    sw   $s0, 0($sp)       # save $s0
    add  $s0, $zero, $zero # i = 0
L1: add  $t1, $s0, $a1     # addr of y[i] in $t1
    lbu  $t2, 0($t1)       # $t2 = y[i]
    add  $t3, $s0, $a0     # addr of x[i] in $t3
    sb   $t2, 0($t3)       # x[i] = y[i]
    beq  $t2, $zero, L2    # exit loop if y[i] == 0
    addi $s0, $s0, 1       # i = i + 1
    j    L1                # next iteration of loop
L2: lw   $s0, 0($sp)       # restore saved $s0
    addi $sp, $sp, 4       # pop 1 item from stack
    jr   $ra               # and return
```

// 9/14