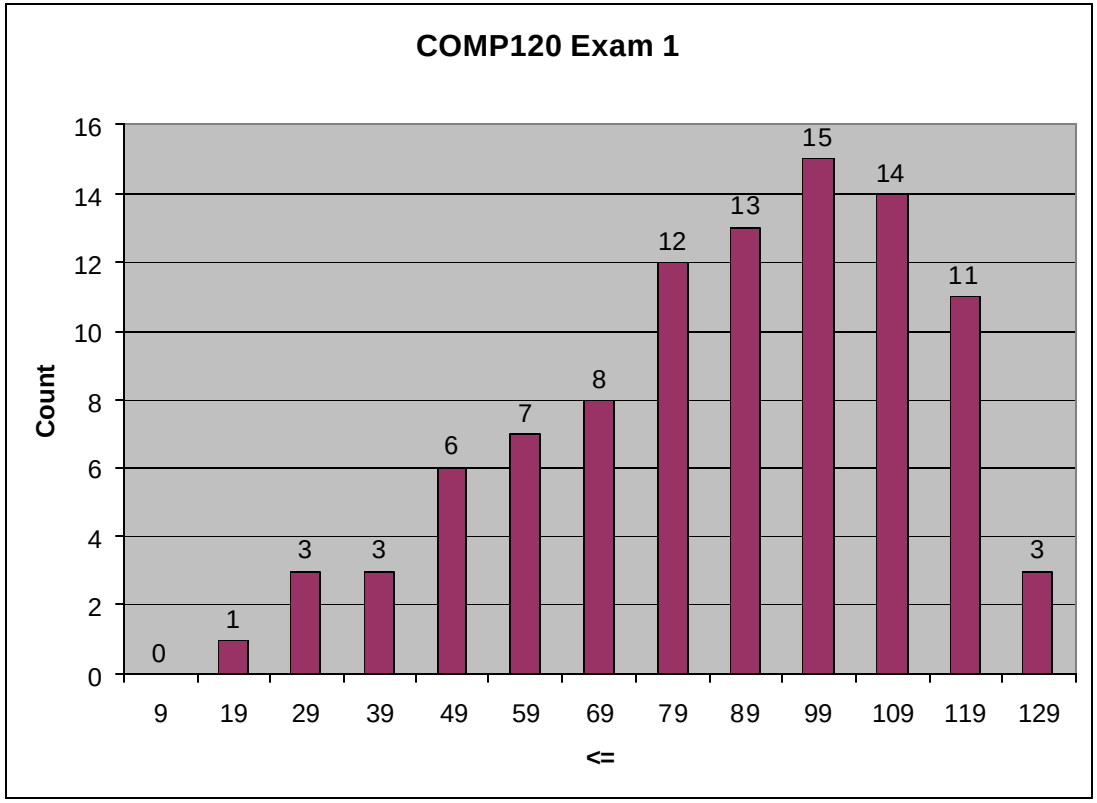
February 15

- Your assignment: ASK QUESTIONS
  - Send email to “gb” describing as carefully as you can what you don’t understand...

Grade Distribution



Mean = 82 points

Median = 87 points

Code Pattern for IF

```
if(COND_EXPR) { STMTS1 } else { STMTS2 }

   CONDCODE(COND_EXPR, Lfalse1, Ltrue2)
Ltrue2:
   CODE(STMTS1)
   j   Lnext3
Lfalse1:
   CODE(STMTS2)
Lnext3:
```

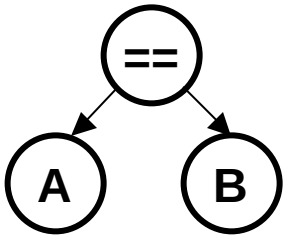
Code Pattern for Conditional Expr

Comparisons: == != < > <= >=

Conjunctions: && ||

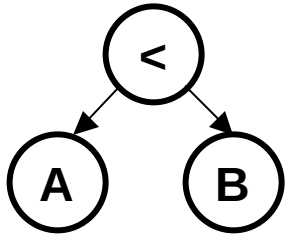
Parenthesis: ( )

A == B




```
        get left in reg A
        get right in reg B
        bne $A,$B, Lfalse
```

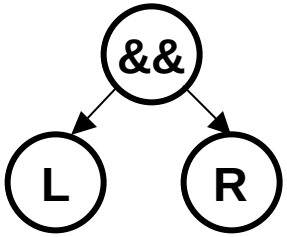
A < B




```
        get left in reg A
        get right in reg B
        slt $t0,A,B
        beq t0,$zero,Lfalse
```

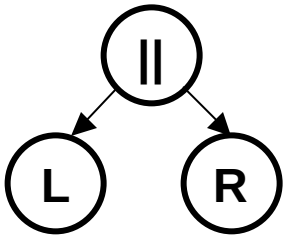
CP for && and ||

L && R




```
        CONDCODE(L, Lfalse, Lnew1)
Lnew1:
        CONDCODE(R, Lfalse, Ltrue)
```

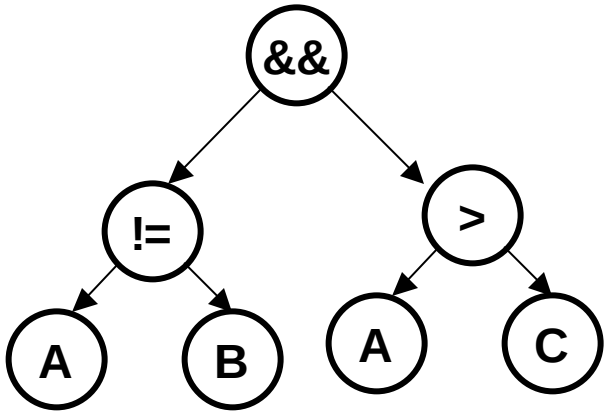
L || R




```
        CONDCODE(L, Lnew1, Ltrue)
Lnew1:
        CONDCODE(R, Lfalse, Ltrue)
```

Example Conditional

if(A != B && A > C) { ST } else { SE }




```
        A=s1, B=s2, C=s3
        beq    $s1,$s2,Lfalse23
Lnew24:
        slt $t0, $s3, $s1
        beq $t0, $zero, Lfalse23
Ltrue25:
        CODE(ST)
        j      Lnext27
Lfalse23:
        CODE(SE)
Lnext27:
```

While

while(COND_EXPR) { STMTS; }

```
Lwhile44:
   CONDCODE(COND_EXPR, Lfalse45, Ltrue46)
Ltrue46:
   CODE(STMTS)
   j   Lwhile44
Lfalse45:
```

FOR

```
for(INIT; COND_EXPR; UPDATE) { STMTS }


   CODE(INIT)
Lfor17:
   CONDCODE(COND_EXPR, Lnext18, Ltrue19)
Ltrue19:
   CODE(STMTS)
   CODE(UPDATE)
   j   Lfor17
Lnext18:
```

Functions

int foo(int a, int b, int c, int d) { STMTS; return EXPR; }

we know a=$a0, b=$a1, c=$a2, d=$a3

assign other simple variables to registers as possible

```
foo:
   save any of $ra or $s0-s7 that are overwritten
   CODE(STMTS)
   CODE(EXPR) leave result in $v0
   restore $ra, and $s0-s7 if we saved them earlier
   jr  $ra
```

cfind

```
int cfind(char str[], char c) {
  for(int i=0; str[i] != 0; i++)
      if(s[i] == c) return i;
  return -1;
}
```

a0 == address of str

a1 == value of character c

v0 == i

no need to save anything

Expand function template

```
cfind:
   # no need to save anything
   CODE('for ...')
   CODE('return -1')
   jr  $ra
```

Expand for

```
cfind:
   CODE('i=0')
Lfor1:
   CONDCODE('s[i] != 0, Lnext2, Ltrue3)
Ltrue3:
   CODE('if ...')
   CODE('i++')
   j   Lfor1
Lnext2:
   CODE('return -1')
   jr  $ra
```

Expand for INIT

```
cfind:
   move      $v0, $zero
Lfor1:
   CONDCODE('s[i] != 0 && s[i] != c', Lnext2, Ltrue3)
Ltrue3:
   CODE('if ...')
   CODE('i++')
   j   Lfor1
Lnext2:
   CODE('return -1')
   jr  $ra
```

Expand for COND_EXPR

```
cfind:
   # no need to save anything
   move      $v0, $zero
Lfor1:
   add $t0, $a0, $v0            # address of s[i]
   lb  $t1, 0($t0)              # t1 = s[i]
   beq $t1, $zero, Lnext2
Ltrue3:
   CODE('if ...')
   CODE('i++')
   j   Lfor1
Lnext2:
   CODE('return -1')
   jr  $ra
```

Expand if

```
cfind:
   move $v0, $zero
Lfor1:
   add  $t0, $a0, $v0        # address of s[i]
   lb   $t1, 0($t0)                 # t1 = s[i]
   beq $t1, $zero, Lnext2
Ltrue3:
   CONDCODE('s[i] == c', Lfalse4, Ltrue5)
Ltrue5:
   CODE('return i');
Lfalse4:
   CODE('i++')
   j    Lfor1
Lnext2:
   CODE('return -1')
   jr   $ra
```

Expand if COND_EXPR

```
cfind:
   move $v0, $zero
Lfor1:
   add  $t0, $a0, $v0        # address of s[i]
   lb   $t1, 0($t0)                 # t1 = s[i]
   beq $t1, $zero, Lnext2
Ltrue3:
   bne $t1, $a1, Lfalse4
Ltrue5:
   CODE('return i');
Lfalse4:
   CODE('i++')
   j    Lfor1
Lnext2:
   CODE('return -1')
   jr   $ra
```

Expand return

```
cfind:
   move $v0, $zero
Lfor1:
   add  $t0, $a0, $v0        # address of s[i]
   lb   $t1, 0($t0)                  # t1 = s[i]
   beq $t1, $zero, Lnext2
Ltrue3:
   bne $t1, $a1, Lfalse4
Ltrue5:
   jr  $ra    # return i already in v0
Lfalse4:
   CODE('i++')
   j    Lfor1
Lnext2:
   CODE('return -1')
   jr   $ra
```

Expand i++

```
cfind:
   move $v0, $zero
Lfor1:
   add  $t0, $a0, $v0        # address of s[i]
   lb   $t1, 0($t0)                 # t1 = s[i]
   beq $t1, $zero, Lnext2
Ltrue3:
   bne $t1, $a1, Lfalse4
Ltrue5:
   jr   $ra    # return i already in v0
Lfalse4:
   addi       $v0, $v0, 1        # i = i+1;
   j    Lfor1
Lnext2:
   CODE('return -1')
   jr   $ra
```

Expand return -1

```
cfind:
   move $v0, $zero
Lfor1:
   add  $t0, $a0, $v0        # address of s[i]
   lb   $t1, 0($t0)                 # t1 = s[i]
   beq  $t1, $zero, Lnext2
Ltrue3:
   bne  $t1, $a1, Lfalse4
Ltrue5:
   jr   $ra    # return i already in v0
Lfalse4:
   addi $v0, $v0, 1          # i = i+1;
   j    Lfor1
Lnext2:
   subi       $v0, $zero, 1     # return value = -1
   jr   $ra
```

Remove unused labels

```
cfind:
   move      $v0, $zero         # i = 0
Lfor1:
   add $t0, $a0, $v0            # address of s[i]
   lb  $t1, 0($t0)              # t1 = s[i]
   beq $t1, $zero, Lnext2       # if s[i] == 0
   bne $t1, $a1, Lfalse4        # if s[i] == c
   jr  $ra   # return i already in v0
Lfalse4:
   addi      $v0, $v0, 1        # i = i+1;
   j   Lfor1
Lnext2:
   subu      $v0, $zero, 1      # return -1
   jr  $ra
```