5.19 Assembly program example: Subroutines

Assembly program

Subroutines enable programmers to write modular assembly programs. A subroutine has well-defined input and output, so can focus on developing a particular subroutine (or module) independently of other subroutines. Each subroutine should ha recognizable behavior, and the main behavior of the program should be easily understandable via a sequence of subroutine

PARTICIPATION ACTIVITY 5.19.1: Modular program development with subroutine: Calculating employee pay.

Start 2x speed

Desired behavior: Calculate employee pay with overtime.

```
# Load hours worked from DM
# Calculate overtime hours
jal CalcOvertimeHours
# Calculate pay
jal CalcPay
# Store pay to DM
```

Main program loads hours worked from memory, calls subroutines to calculate overtime hours and pay, and stores pay to memory.

```
CalcOvertimeHours:
   # Subroutine instructions
   jr $ra
```

CalcOvertimeHours calculates overtime hours (hours greater than 40) given total hours worked.

```
CalcPay:
   # Subroutine instructions
   jr $ra
```

CalcPay calculates pay given total hours worked, overtime hours, and hourly wage.

PARTICIPATION ACTIVITY 5.19.2: Modular program development.

Refer to the animation above.

1) Modular development means to divide a program into separate modules (or subroutines) that can be developed and tested separately.
   - ○ True
   - ○ False

2) The main program behavior only consists of jal instructions to call subroutines.
   - ○ True
   - ○ False

3) The CalcPay subroutine can be written before the CalcOvertimeHours subroutine.
   - ○ True
   - ○ False

Modular subroutine development

The subroutines for calculating the number of overtime hours and calculating the employee's pay can be developed separa

Overtime is the number of hours worked beyond 40 hours in a single week. Ex: If an employee works 55 hours, the employe hours of overtime. If an employee works 40 hours or fewer , then the employee worked zero overtime hours. The CalcOvert

subroutine below calculates the number of overtime hours an employee has worked given the number of total hours the er in a week. $t0 is used for the subroutine's argument, which is the number of hours worked in a week. $t1 is used for the sul value, which is the number of overtime hours.

Figure 5.19.1: CalcOvertimeHours subroutine calculates an employee's overtime hours.

```
CalcOvertimeHours:
   addi $t2, $zero, 40
   slt $t3, $t0, $t2
   bne $t3, $zero, NoOvertime
   # Overtime worked
   # Overtime hours is 40 - hours worked
   sub $t1, $t0, $t2
   j ReturnOvertime
NoOvertime:
   # No overtime, so overtime hours is 0
   addi $t1, $zero, 0
ReturnOvertime:
   jr $ra
```

PARTICIPATION ACTIVITY 5.19.3: CalcOvertimeHours.

1) If $t0 holds 35, what is $t1 after the CalcOvertimeHours subroutines returns?
   - ○ 0
   - ○ 35
   - ○ 40

2) If $t0 holds 42, what is $t1 after the CalcOvertimeHours subroutines returns?
   - ○ 0

- 2
- 40

3) If the employee did not work overtime, which instruction writes 0 to the register for the return value?
   - addi $t2, $zero, 40
   - addi $t1, $zero, 0
   - sub $t1, $t0, $t2

An employee is paid an hourly wage for the first 40 hours, and two times the hourly wage for overtime hours. The CalcPay s calculates an employee's weekly pay. An employee's hourly pay rate is $10/hour. The subroutine's uses $t0 for the total hou for the employee hourly wage, and $t2 for the number of overtime hours. The subroutine returns the employee's pay using

Figure 5.19.2: CalcPay subroutine calculates an employee's pay.

```
CalcPay:
   # Calculate base pay
   mul $t3, $t0, $t1
   # Calculate overtime pay
   mul $t4, $t2, $t1
   # Calculate total pay
   add $t3, $t3, $t4
   jr $ra
```

PARTICIPATION ACTIVITY 5.19.4: CalcPay subroutine.

1) Which register is used for the hourly wage?
   - $t1

- $t2

2) The total pay is calculated as the total hours times the hourly wage plus the overtime hours times the hourly wage.
   - True
   - False

3) If $t0 holds 30, $t1 holds 10, and $t2 holds 0, what is $t3 after the subroutine returns?
   - 300
   - 600

4) If $t0 holds 55, $t1 holds 10, and $t2 holds 15, what is $t3 after the subroutine returns?
   - 550
   - 700

Main program behavior is a sequence of subroutine calls.

The program below calculates the pay for a single employee, where DM[5000] is the total hours worked by the employee, a is stored to DM[5040]. The program's main behavior consists of loading the hours worked, calling the CalcOvertimeHours s calculate the overtime hours, calling the CalcPay subroutine to calculate the pay, and storing the pay to memory.

Figure 5.19.3: Calculating pay for a single employee.

```
# Load hours worked from DM[5000]
addi $t6, $zero, 5000
lw $t0, 0($t6)
jal CalcOvertimeHours
# Overtime hours returned in $t1
# Copy $t1 to $t2
add $t2, $zero, $t1
# Initialize pay rate to $10/hour
addi $t1, $zero, 10
jal CalcPay
# Pay is returned in $t3
# Store pay to DM[5040]
addi $t6, $zero, 5040
sw $t3, 0($t6)
j Done

CalcOvertimeHours:
   addi $t2, $zero, 40
   slt $t3, $t0, $t2
   bne $t3, $zero, NoOvertime
   # Overtime worked
   # Overtime hours is 40 - hours worked
   sub $t1, $t0, $t2
   j ReturnOvertime
NoOvertime:
   # No overtime, so overtime hours is 0
   addi $t1, $zero, 0
ReturnOvertime:
   jr $ra

CalcPay:
   # Calculate base pay
   mul $t3, $t0, $t1
   # Calculate overtime pay
   mul $t4, $t2, $t1
   # Calculate total pay
   add $t3, $t3, $t4
   jr $ra

Done:
```

PARTICIPATION ACTIVITY 5.19.5: Using the subroutines to calculate employee pay.

Refer to the program above.

1) What does `add $t2, $zero, $t1` do?
   - ○ Initializes the total pay.
   - ○ Copies $t1 to $t2.

2) Which instruction writes the total hours worked argument for CalcPay?
   - ○ addi $t1, $zero, 10
   - ○ add $t2, $zero, $t1
   - ○ lw $t0, 0($t6)

3) What does `j Done` do?
   - ○ Calls the done subroutine.
   - ○ Jumps past the CalcOvertimeHour and CalcPay subroutines.

PARTICIPATION ACTIVITY 5.19.6: Calculating pay for multiple employees.

Extend the program below to calculate pay for multiple employees, where each employee has a different hourly wage.

1. Modify the program to calculate pay for three employees. DM[5000], DM[5004], and DM[5008] are the total hours worked for the three employees. Store the employees' pay to DM[5040], DM[5044], and DM[5048], respectively.
2. Modify the program to load the employees' pay rates from DM[5020], DM[5024], and DM[5028], respectively.

Assembly

```
 Line 1  # Load hours worked from DM[5000]
 Line 2  addi $t6, $zero, 5000
 Line 3  lw $t0, 0($t6)
 Line 4  jal CalcOvertimeHours
 Line 5  # Overtime hours returned in $t1
 Line 6  # Copy $t1 to $t2
 Line 7  add $t2, $zero, $t1
 Line 8  # Initialize pay rate to $10/hour
 Line 9  addi $t1, $zero, 10
Line 10  jal CalcPay
Line 11  # Pay is returned in $t3
Line 12  # Store pay to DM[5040]
Line 13  addi $t6, $zero, 5040
Line 14  sw $t3, 0($t6)
Line 15  j Done
Line 16
Line 17  CalcOvertimeHours:
Line 18     addi $t2, $zero, 40
```

Registers

$zero	0
$t0	0
$t1	0
$t2	0
$t3	0
$t4	0
$t5	0
$t6	0

+

5000	
5004	
5008	
5020	
5024	
5028	
5040	
5044	
5048	

+

ENTER SIMULATION STEP RUN

More options

Provide feedback on this section