4) Flow control

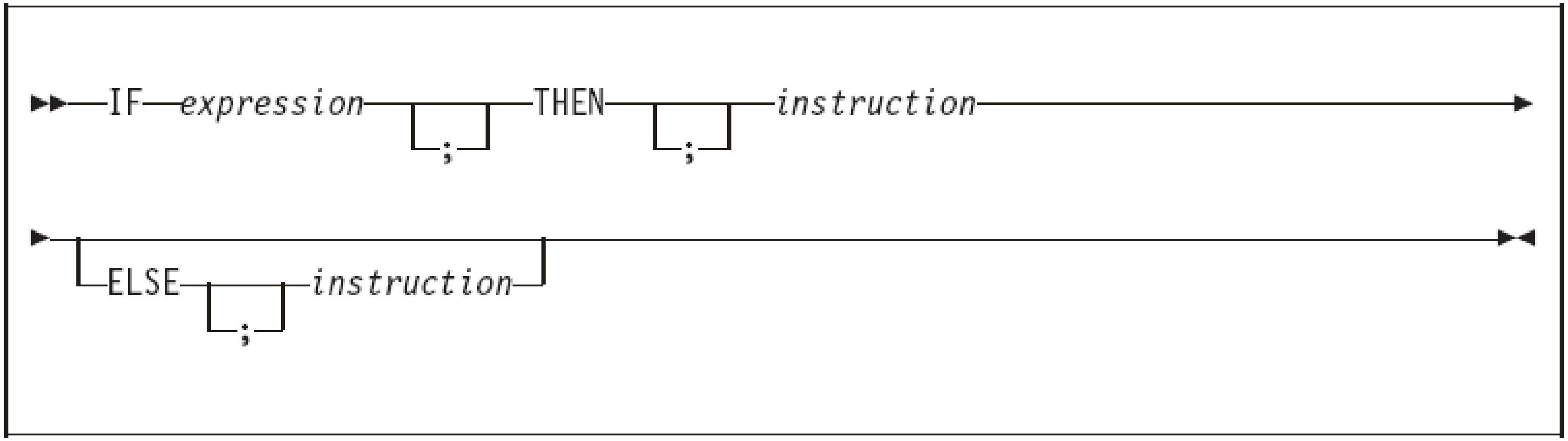
- Logical operators,
- comparative operators

Instructions:

- IF/THEN/ELSE,
- DO/END – composite instruction.
- SELECT,
- NOP.

Resources: TSO REXX Reference

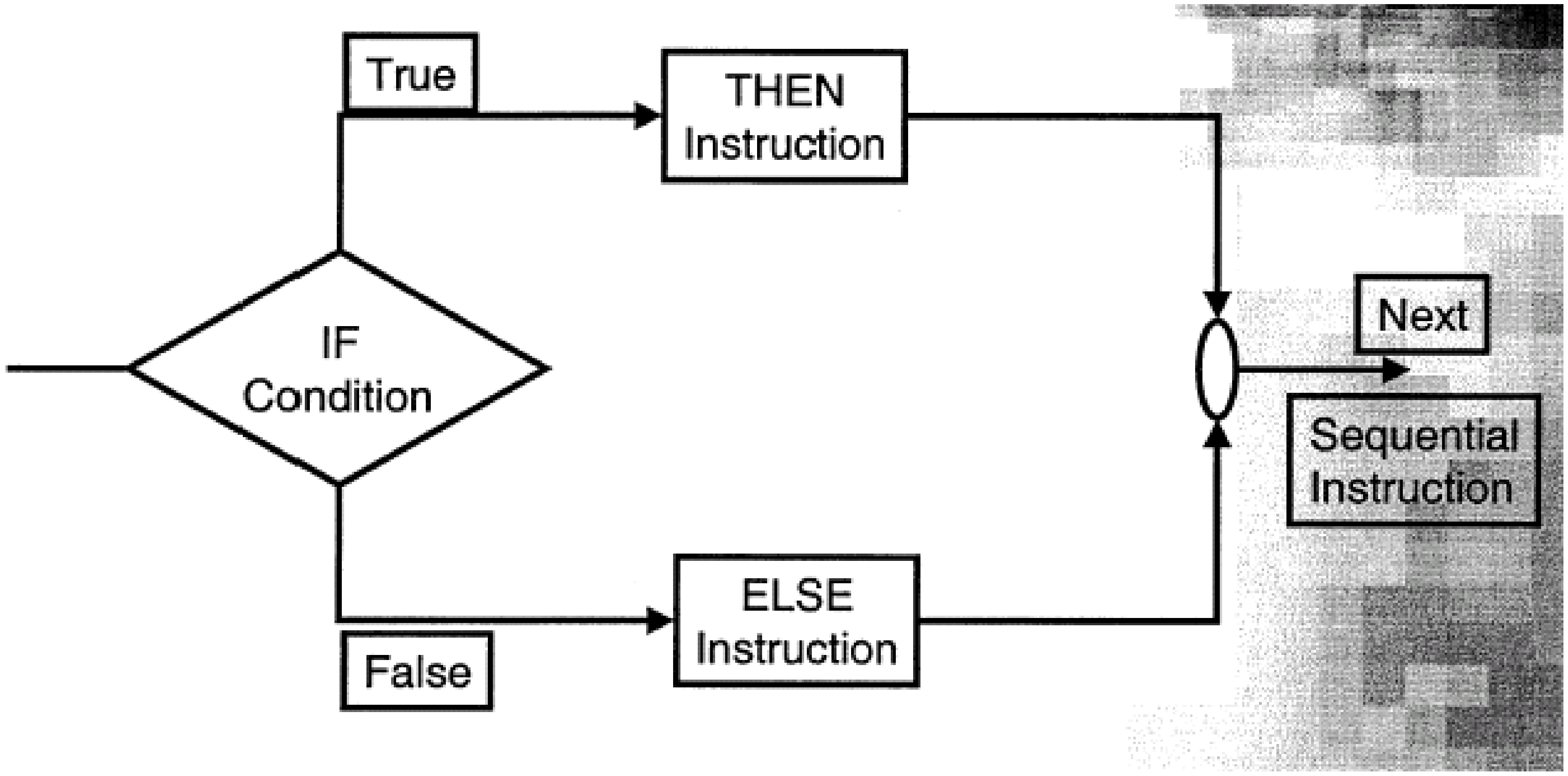
IF/THEN/ELSE



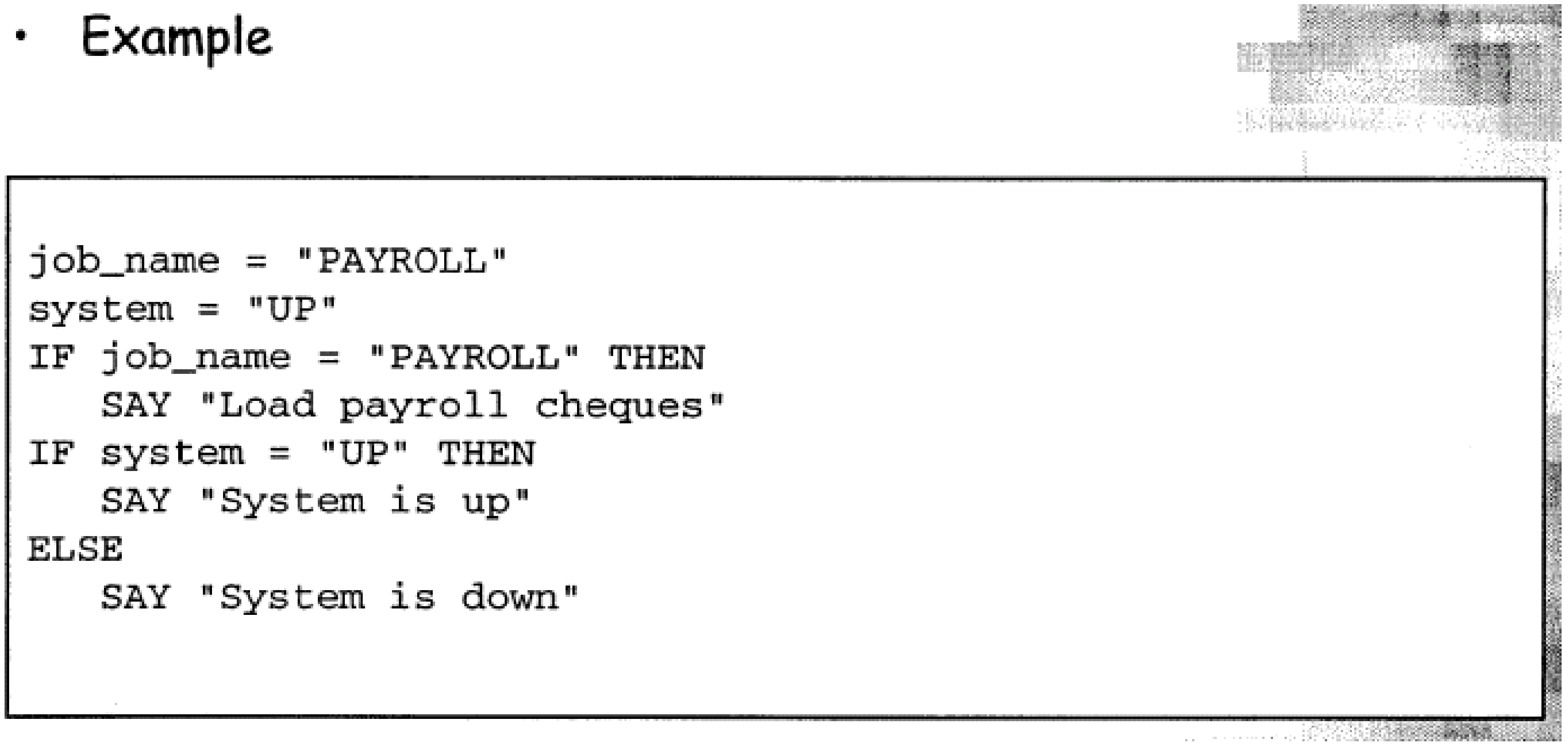
IF conditionally processes an instruction or group of instructions depending on the evaluation of the *expression*. The *expression* is evaluated and must result in `0` or `1`.

The instruction after the THEN is processed only if the result is `1` (true). If you specify an ELSE, the instruction after the ELSE is processed only if the result of the evaluation is `0` (false).

IF/THEN/ELSE


True
THEN Instruction
IF Condition
Next
Sequential Instruction
ELSE Instruction
False

IF/THEN/ELSE



- Example

```
job_name = "PAYROLL"
system = "UP"
IF job_name = "PAYROLL" THEN
   SAY "Load payroll cheques"
IF system = "UP" THEN
   SAY "System is up"
ELSE
   SAY "System is down"
```

Test Exercise 41

- Write a REXX program to test the statements below.
- Correct the code where necessary

```
test_value = 1
IF test_value = 1 SAY "Yes"
IF test_value = 1
   THEN SAY "Yes"
IF test_value = 1 THEN SAY "Yes" ELSE SAY "No"
IF test_value = 1 ELSE SAY "No"
IF test_value = 1
   THEN
      SAY "Yes"
   ELSE
      SAY "No"
```

Nested IF statement

- IF statements also may be nested within IF statements

```
PARSE ARG age
IF age < 65 THEN
   IF age > 21 THEN
      SAY "Over 21 and under 65"
   ELSE
      IF age >= 16 THEN
         SAY "Between 16 and 21"
      ELSE
         SAY "Under 16"
ELSE
   SAY "65 or over"
```

Simple DO END

- This groups several statements together so that REXX will treat them as one instruction

- Often you need to execute more than one instruction in a THEN or ELSE clause

```
system_state = "UP"
IF system_state = "UP" THEN DO
   SAY "The system should be down"
   system_state = "DOWN"
END
```

Comparative operators

- Compare two terms and return 1 if the result is true and 0 if then result is false

- Normal comparison
  - = equal
  - \\= not equal (can also use not sign, X'5F')
  - > greater than
  - < less than
  - >< greater than or less than (same as not equal)
  - >= greater than or equal to
  - <= less than or equal to
  - \\< not less than
  - \\> no greater than

Comparative operators - sample

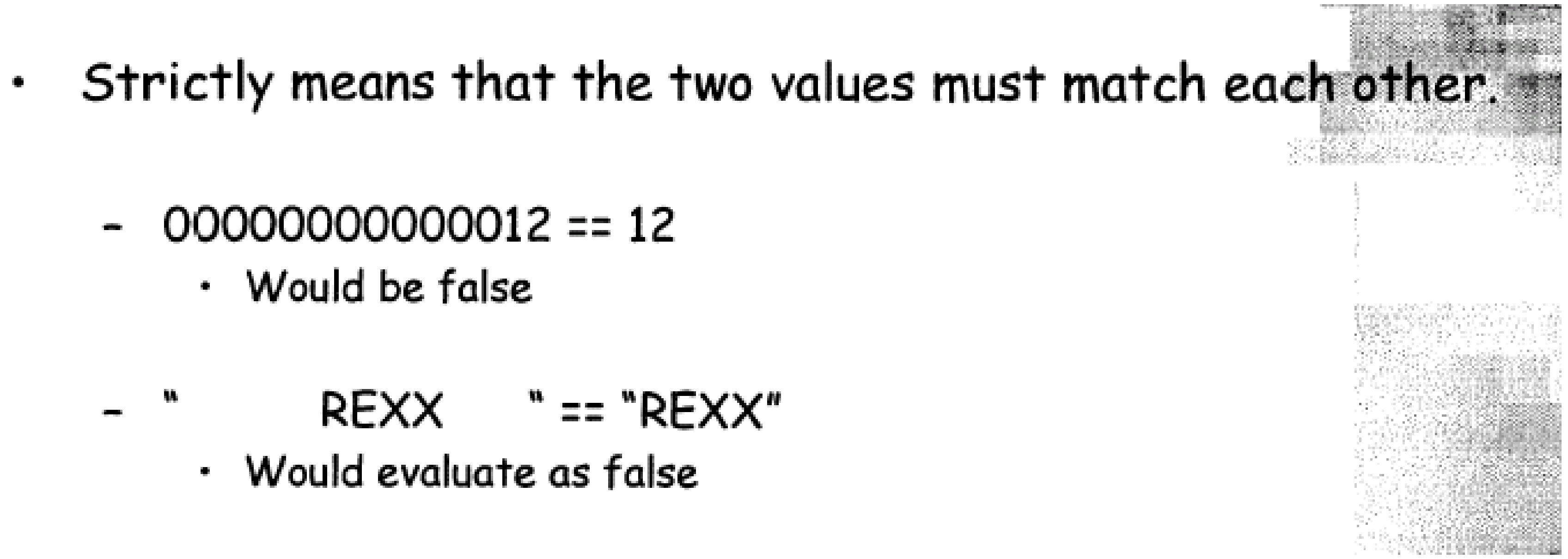
- When REXX compares two non-numeric values, it ignores leading and trailing spaces
  - "   REXX   " = "REXX"
    - Would evaluate as true
- When REXX compares two numeric values it ignores leading and trailing zeros.
  - 0000000012 = 12
  - 12 = 12.000
    - Would evaluate to true

Comparative operators - strict comparison

==	True if terms are strictly equal (identical)
\==, ¬==, /==	True if the terms are NOT strictly equal (inverse of ==)
>>	Strictly greater than
<<	Strictly less than
>>=	Strictly greater than or equal to
\<<, ¬<<	Strictly NOT less than
<<=	Strictly less than or equal to
\>>, ¬>>	Strictly NOT greater than

Guideline: Throughout the language, the **not** character, ¬, is synonymous with the backslash (\). You can use the two characters interchangeably, according to

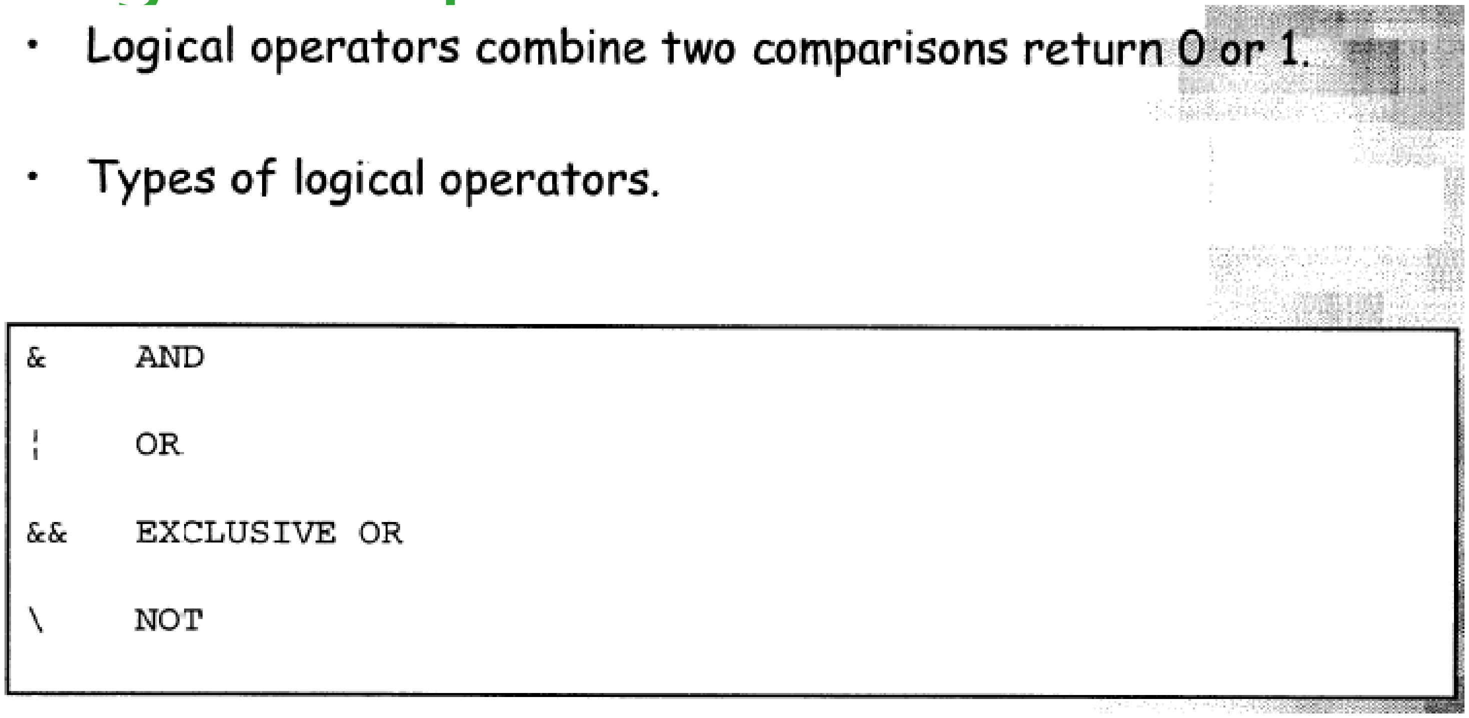
Comparative operators - strict comparison - sample



- Strictly means that the two values must match each other.
  - 0000000000012 == 12
    - Would be false
  - "   REXX   " == "REXX"
    - Would evaluate as false

This course has been prepared by Milos Forman for MCoE needs only!

Logical Operator



- Logical operators combine two comparisons return 0 or 1.
- Types of logical operators.

&	AND
¦	OR
&&	EXCLUSIVE OR
\	NOT

Priorities:
Arithmetic operators
Concatenation operators
Comparative
Logical operators

\
&
|, &&

Logical Operator

& AND - returns a 1 (true) if both comparisons are true, and a 0 (false) otherwise - performs a logical AND operation

| OR - returns a 1 (true) if at least one comparison of several is true, and a 0 (false) otherwise - performs a logical or operation

&& EXCLUSIVE OR - returns a 1 (true) if ONLY one of a group of comparisons is true, and a 0 (false) otherwise - performs a logical exclusive OR function

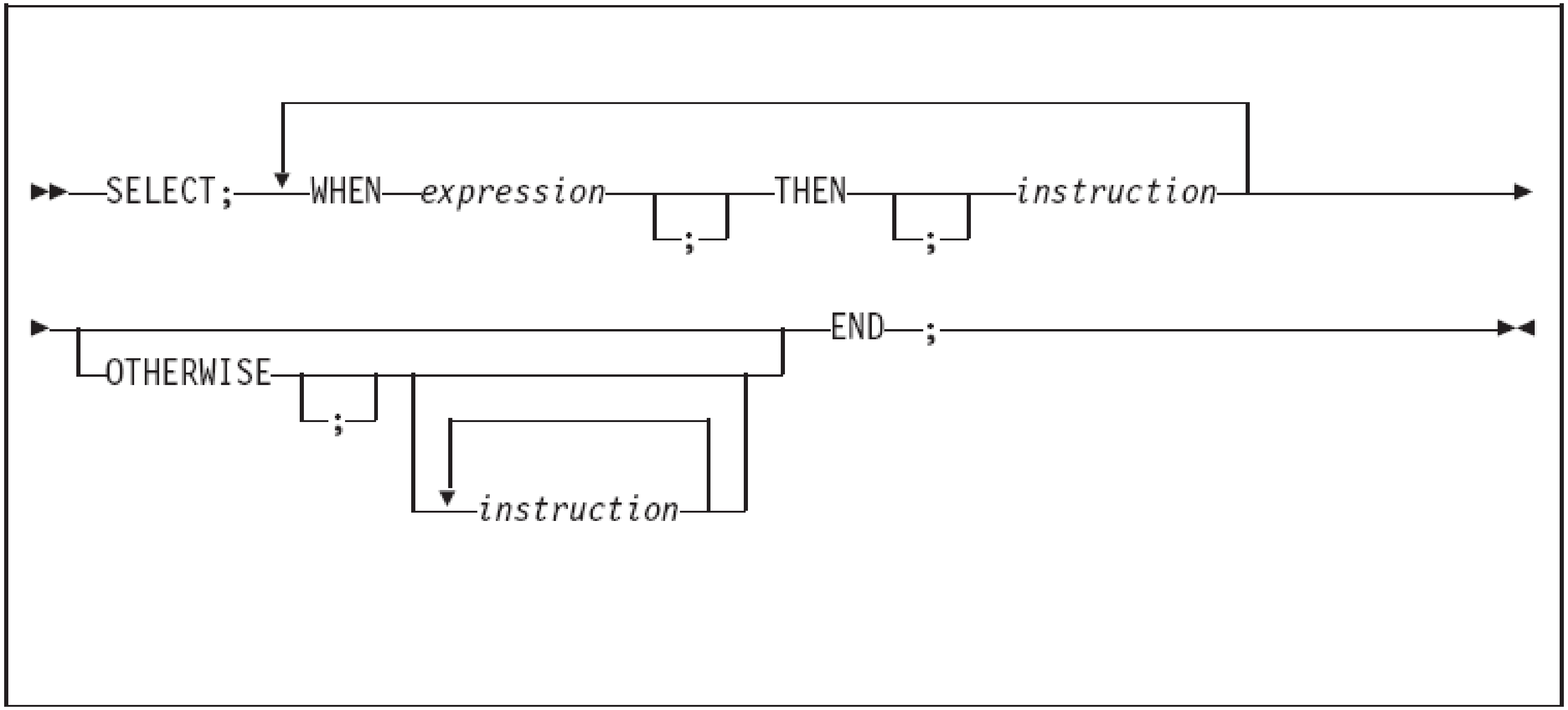
\ NOT - returns the reverse logical value for an expression returns false if expression resolves to true, and true if the expression resolves to false

Multiple Logical Operator

- When multiple logical operators are used, &s are evaluated before |s.

```
test_value = 1
old_value = 2000
new_value = 3
IF test_value = 1 & (old_value = 2 | new_value = 3) THEN
   SAY "All ok"
```

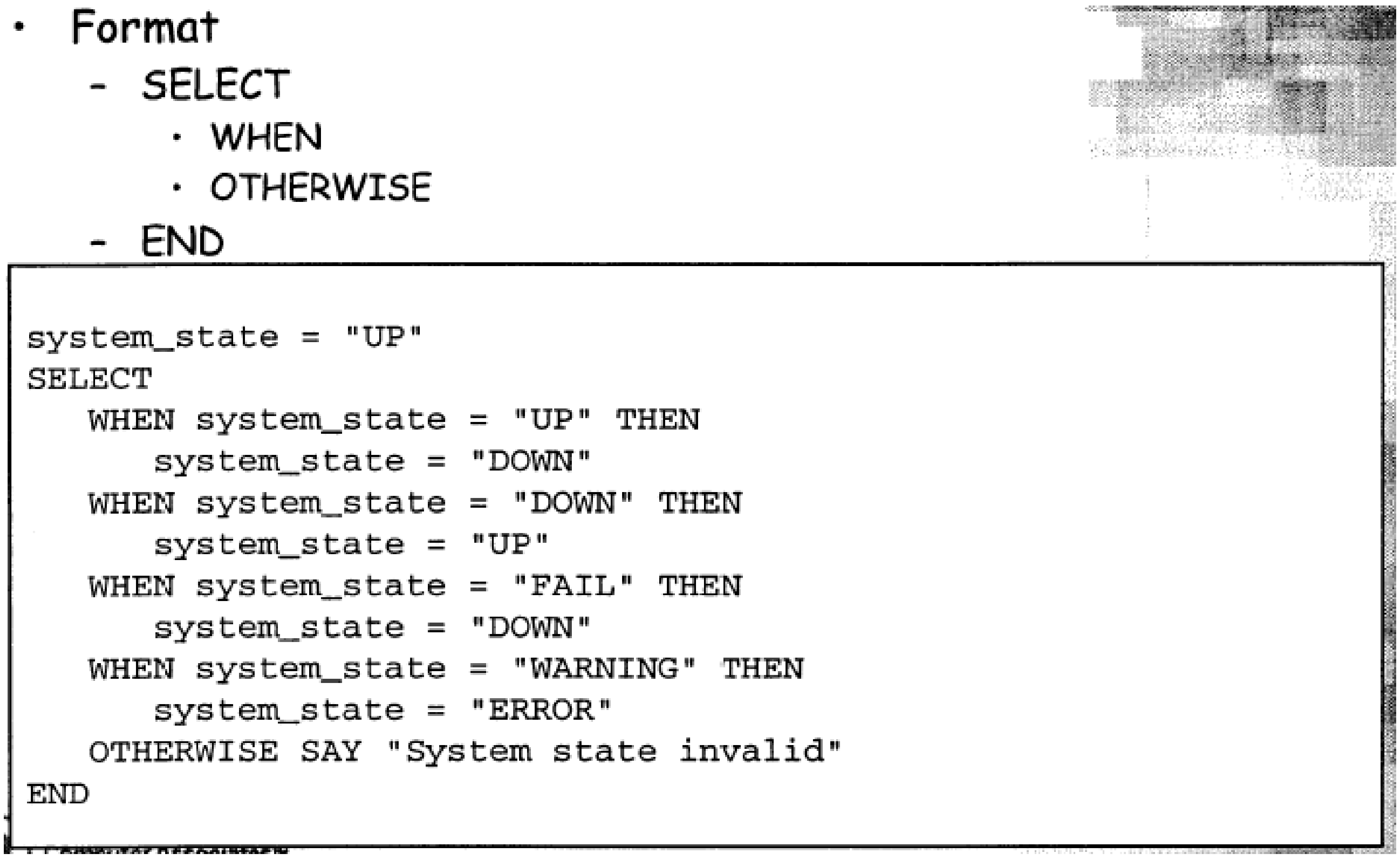
SELECT



SELECT conditionally calls one of several alternative instructions.

Unlike IF with ELSE, the SELECT statement requires the OTHERWISE for all false conditions.

SELECT Sample



- Format
  - SELECT
    - WHEN
    - OTHERWISE
  - END

```
system_state = "UP"
SELECT
   WHEN system_state = "UP" THEN
      system_state = "DOWN"
   WHEN system_state = "DOWN" THEN
      system_state = "UP"
   WHEN system_state = "FAIL" THEN
      system_state = "DOWN"
   WHEN system_state = "WARNING" THEN
      system_state = "ERROR"
   OTHERWISE SAY "System state invalid"
END
```

NOP

- Dummy instruction that has no effect
- Often used with and IF and SELECT

```
system_state = "UP"
SELECT
   WHEN system_state = "UP" THEN
      system_state = "DOWN"
   WHEN system_state = "DOWN" THEN
      system_state = "UP"
   WHEN system_state = "FAIL" THEN
      system_state = "DOWN"
   WHEN system_state = "WARNING" THEN
      system_state = "ERROR"
   OTHERWISE NOP
END
```

Work section 4.1

- Re-Write the "AGE" nested IF as four separate IF statements in order to perform the same function, in a REXX program. Assign the value AGE as an argument.

```
ex 'clcs.iulc00.rexx(rx10141)' '65'

65 or over
***
```

AGE =	Result
10	
21	
65	
70	
Your age	

This course has been prepared by Milos Forman for MCoE needs only!

Work section 4.1 (Continued)

```
IF age < 65 THEN
   IF age > 21 THEN
      SAY "Over 21 and under 65"
   ELSE
      IF age >= 16 THEN
         SAY "Between 16 and 21"
      ELSE
          SAY "Under 16"
ELSE
   SAY "65 or over”
```

Enter the results in the table above.

Hint do not use any else statements.

Work section 4.2

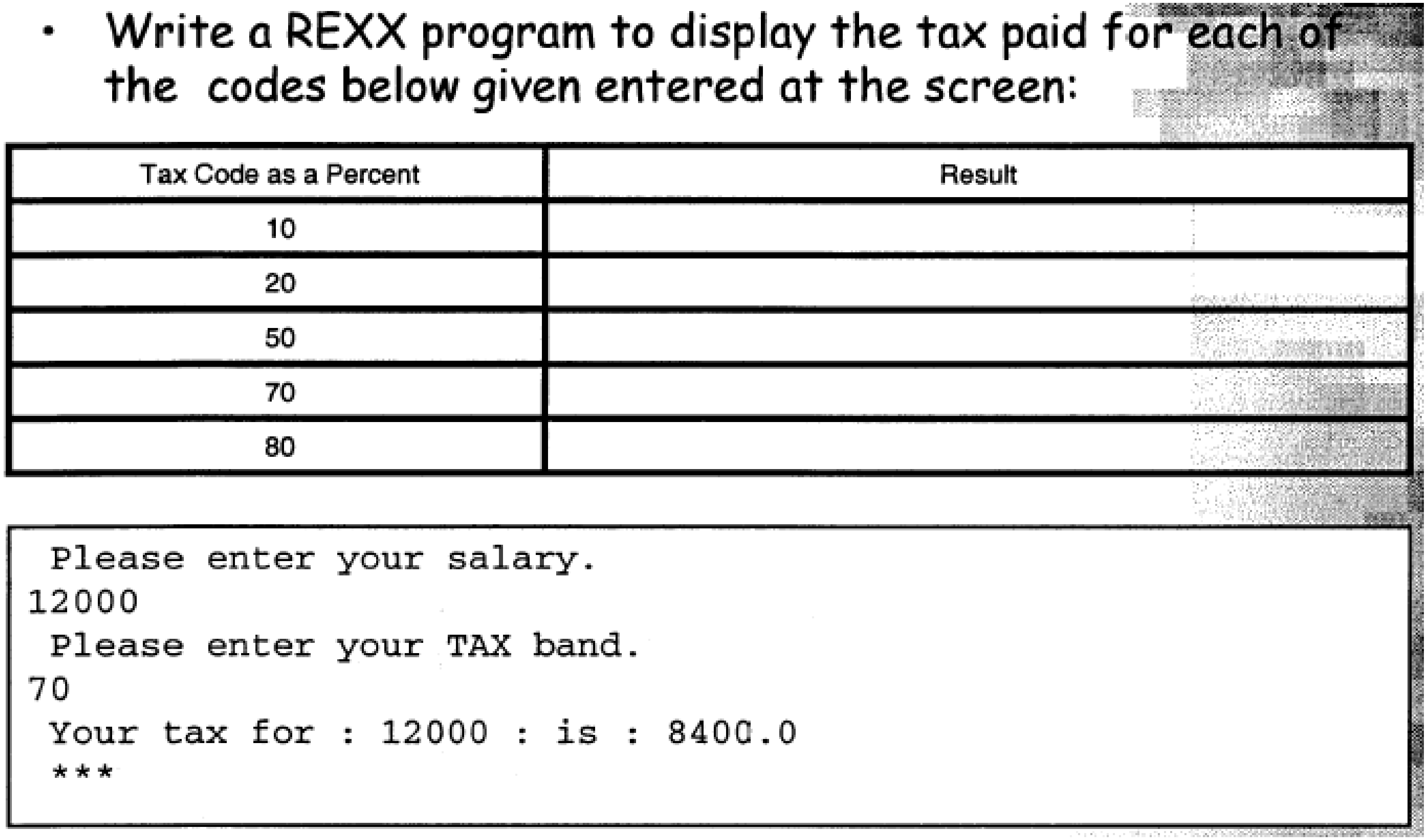
- Re-Write Work section 4.1 "AGE" Using the select Statement

```
ex 'clcs.iulc00.rexx(rx10131)' '65'

65 or over
***
```

AGE =	Result
10	
21	
65	
70	
Your age	

Additional Program



- Write a REXX program to display the tax paid for each of the codes below given entered at the screen:

Tax Code as a Percent	Result
10	
20	
50	
70	
80	

```
 Please enter your salary.
12000
 Please enter your TAX band.
70
 Your tax for : 12000 : is : 8400.0
 ***
```

This course has been prepared by Milos Forman for MCoE needs only!

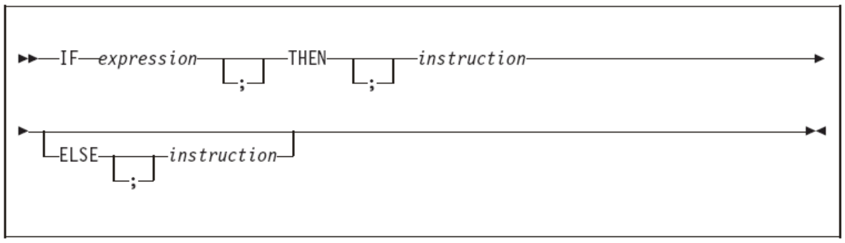
4) Flow control

- Logical operators,
- comparative operators

Instructions:
- IF/THEN/ELSE,
- DO/END – composite instruction.
- SELECT,
- NOP.

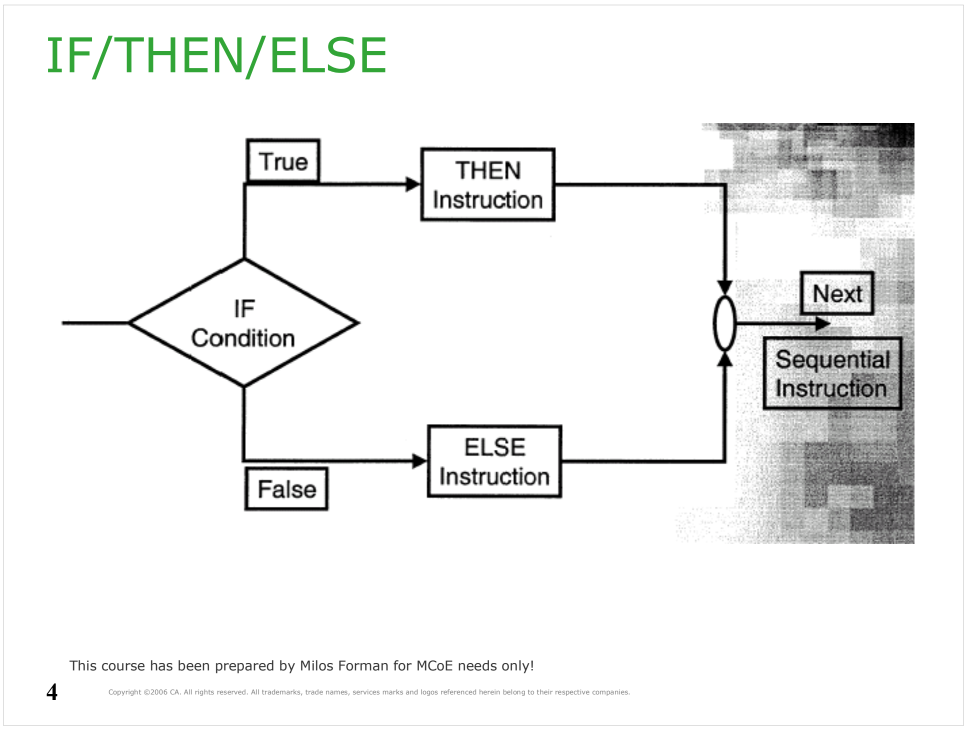
Resources: TSO REXX Reference

IF/THEN/ELSE

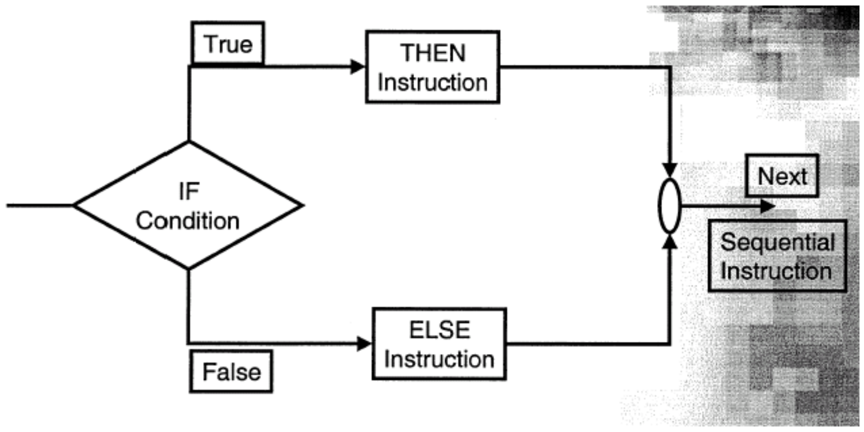


IF conditionally processes an instruction or group of instructions depending on the evaluation of the *expression*. The *expression* is evaluated and must result in `0` or `1`.

The instruction after the THEN is processed only if the result is `1` (true). If you specify an ELSE, the instruction after the ELSE is processed only if the result of the evaluation is `0` (false).



IF/THEN/ELSE

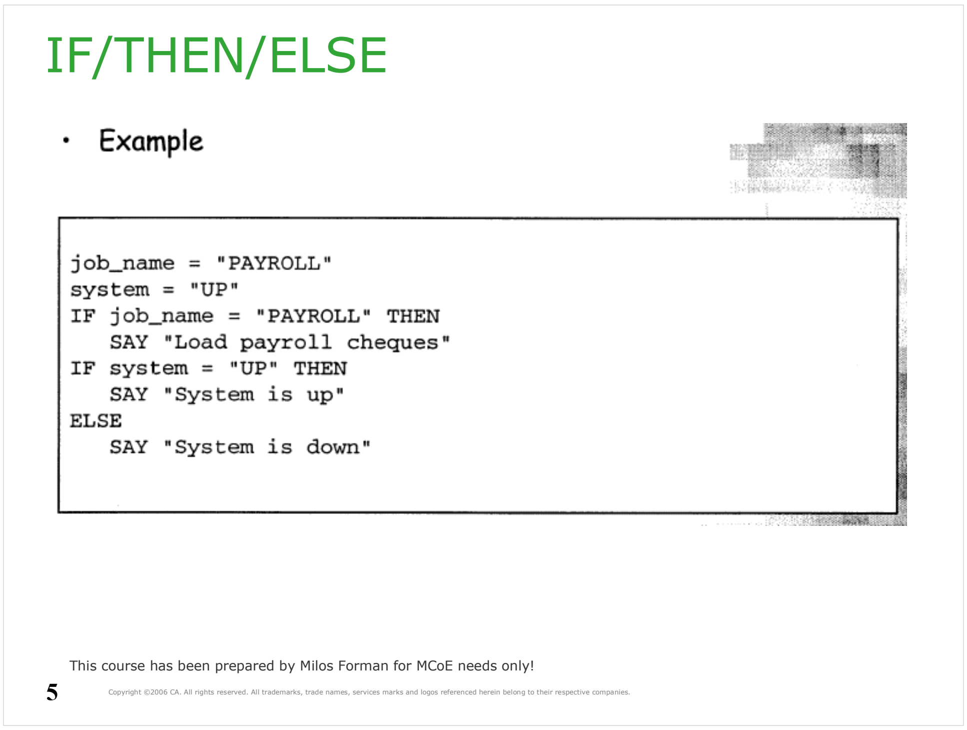


This course has been prepared by Milos Forman for MCoE needs only!

•Expression must evaluate to 1 or 0 (true or false).

•Every IF must have THEN and instruction indicating what to do if the expression is true.

•The ELSE clause indicates what to do if the expression is false, optional.



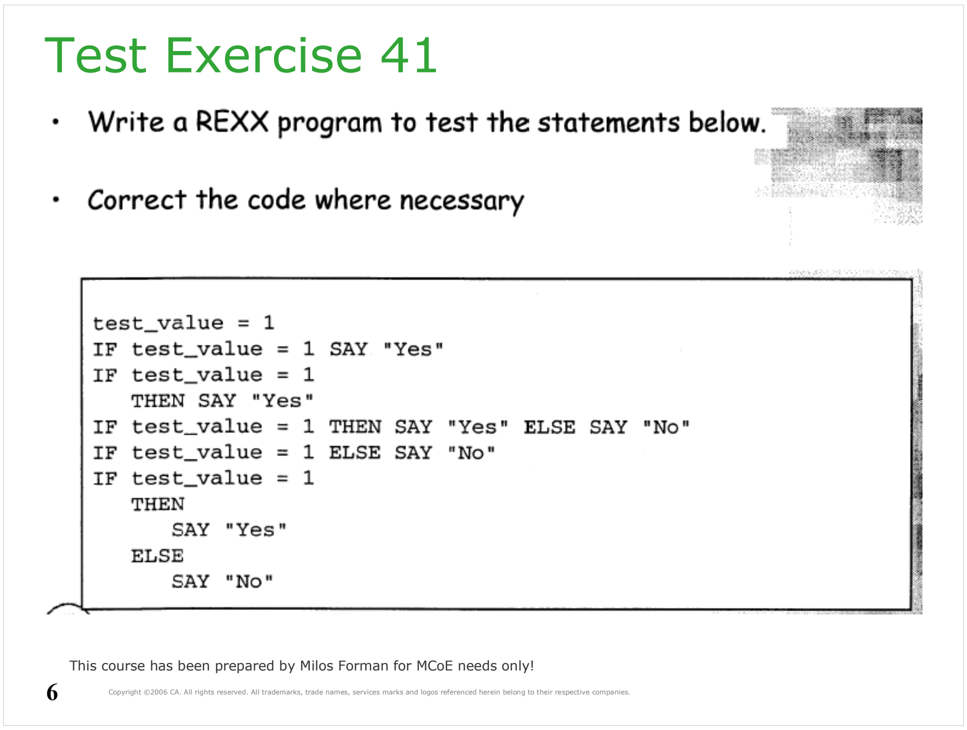
IF/THEN/ELSE

- Example

```
job_name = "PAYROLL"
system = "UP"
IF job_name = "PAYROLL" THEN
   SAY "Load payroll cheques"
IF system = "UP" THEN
   SAY "System is up"
ELSE
   SAY "System is down"
```

5

Write it and test it.



Test Exercise 41

- Write a REXX program to test the statements below.
- Correct the code where necessary

```
test_value = 1
IF test_value = 1 SAY "Yes"
IF test_value = 1
   THEN SAY "Yes"
IF test_value = 1 THEN SAY "Yes" ELSE SAY "No"
IF test_value = 1 ELSE SAY "No"
IF test_value = 1
   THEN
      SAY "Yes"
   ELSE
      SAY "No"
```

This course has been prepared by Milos Forman for MCoE needs only!

6

Write it and test it.

then is missing on the second line and **then say “Yes”** on the sixth line.

Nested IF statement

- IF statements also may be nested within IF statements

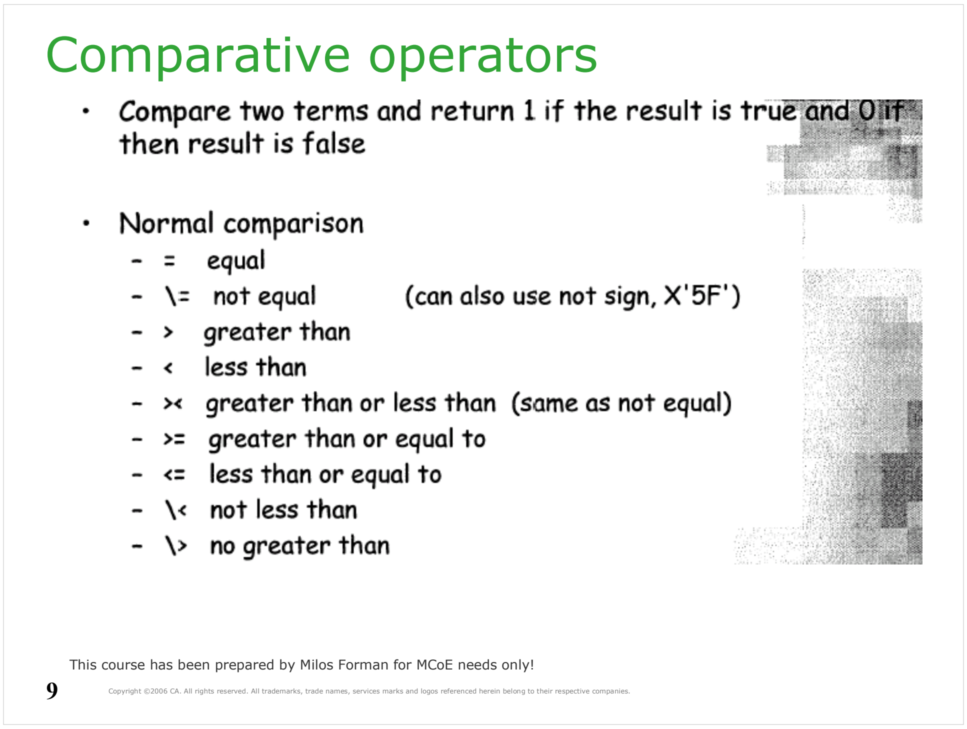
```
PARSE ARG age
IF age < 65 THEN
   IF age > 21 THEN
      SAY "Over 21 and under 65"
   ELSE
      IF age >= 16 THEN
         SAY "Between 16 and 21"
      ELSE
         SAY "Under 16"
ELSE
   SAY "65 or over"
```

Simple DO END

- This groups several statements together so that REXX will treat them as one instruction
- Often you need to execute more than one instruction in a THEN or ELSE clause

```
system_state = "UP"
IF system_state = "UP" THEN DO
   SAY "The system should be down"
   system_state = "DOWN"
END
```

This course has been prepared by Milos Forman for MCoE needs only!



Comparative operators

- Compare two terms and return 1 if the result is true and 0 if then result is false
- Normal comparison
  - = equal
  - \= not equal (can also use not sign, X'5F')
  - > greater than
  - < less than
  - >< greater than or less than (same as not equal)
  - >= greater than or equal to
  - <= less than or equal to
  - \< not less than
  - > no greater than

This course has been prepared by Milos Forman for MCoE needs only!

Comparative operators - sample

- When REXX compares two non-numeric values, it ignores leading and trailing spaces
  - “   REXX   “ = “REXX”
    - Would evaluate as true
- When REXX compares two numeric values it ignores leading and trailing zeros.
  - 00000000012 = 12
  - 12 = 12.000
    - Would evaluate to true

Comparative operators - strict comparison

==	True if terms are strictly equal (identical)
\==, ¬==, /==	True if the terms are NOT strictly equal (inverse of ==)
>>	Strictly greater than
<<	Strictly less than
>>=	Strictly greater than or equal to
\<<, ¬<<	Strictly NOT less than
<<=	Strictly less than or equal to
\>>, ¬>>	Strictly NOT greater than

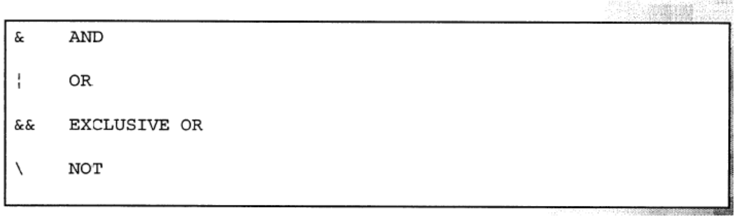
Guideline: Throughout the language, the **not** character, ¬, is synonymous with the backslash (\). You can use the two characters interchangeably, according to

Comparative operators - strict comparison - sample

- Strictly means that the two values must match each other.
  - 00000000000012 == 12
    - Would be false
  - "   REXX   " == "REXX"
    - Would evaluate as false

Logical Operator

- Logical operators combine two comparisons return 0 or 1.
- Types of logical operators.



&	AND
¦	OR
&&	EXCLUSIVE OR
\	NOT

Priorities:
Arithmetic operators
Concatenation operators
Comparative
Logical operators
\
&
|, &&

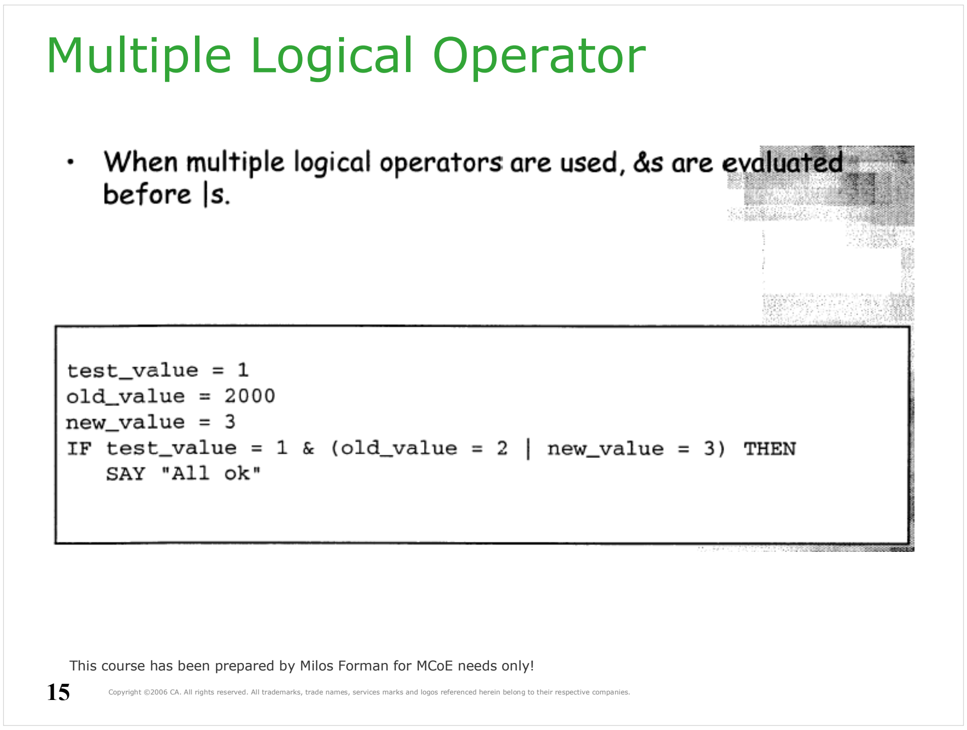
Logical Operator

& AND - returns a 1 (true) if both comparisons are true, and a 0 (false) otherwise - performs a logical AND operation

| OR - returns a 1 (true) if at least one comparison of several is true, and a 0 (false) otherwise - performs a logical or operation

&& EXCLUSIVE OR - returns a 1 (true) if ONLY one of a group of comparisons is true, and a 0 (false) otherwise - performs a logical exclusive OR function

\ NOT - returns the reverse logical value for an expression returns false if expression resolves to true, and true if the expression resolves to false



Multiple Logical Operator

- When multiple logical operators are used, &s are evaluated before |s.

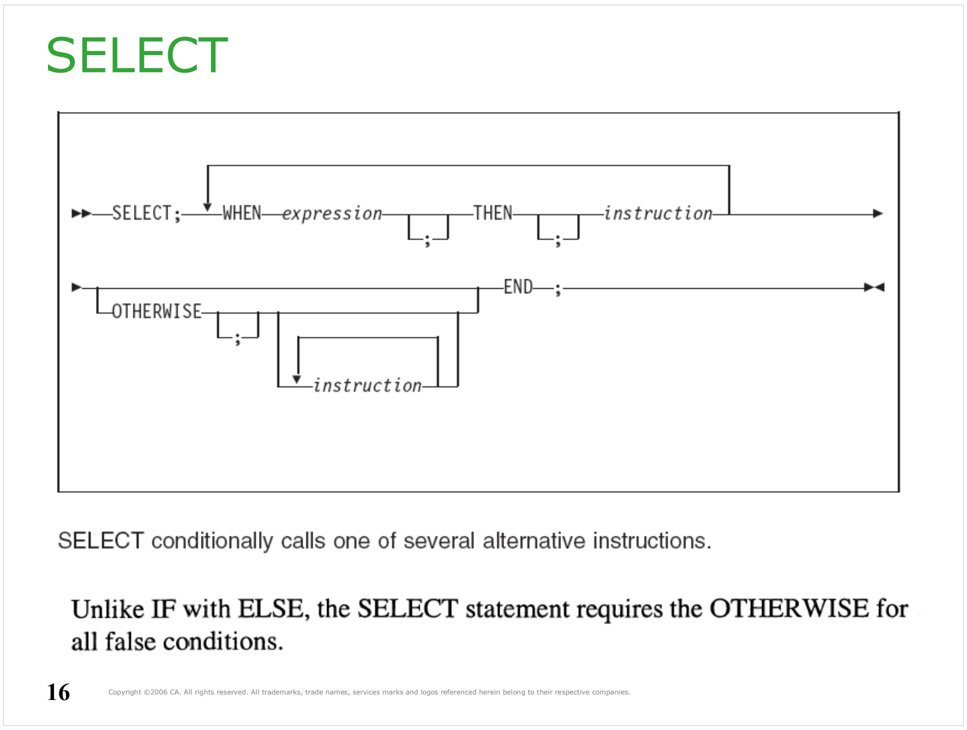
```
test_value = 1
old_value = 2000
new_value = 3
IF test_value = 1 & (old_value = 2 | new_value = 3) THEN
   SAY "All ok"
```

This course has been prepared by Milos Forman for MCoE needs only!

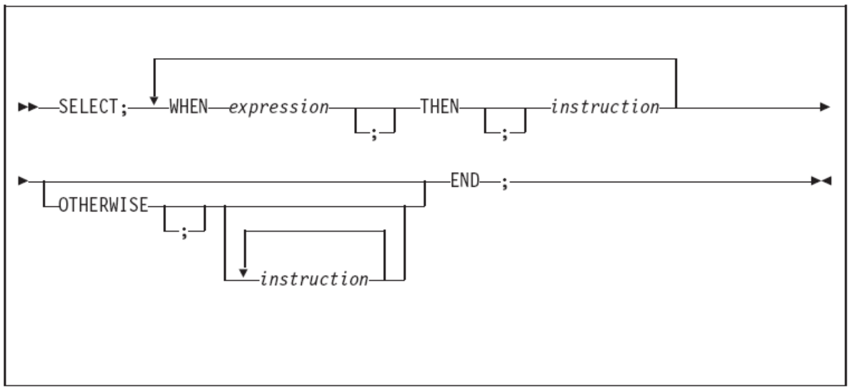
15

Write it and test it.

See ‘MCOE.REXA.REXX(RX201411)’



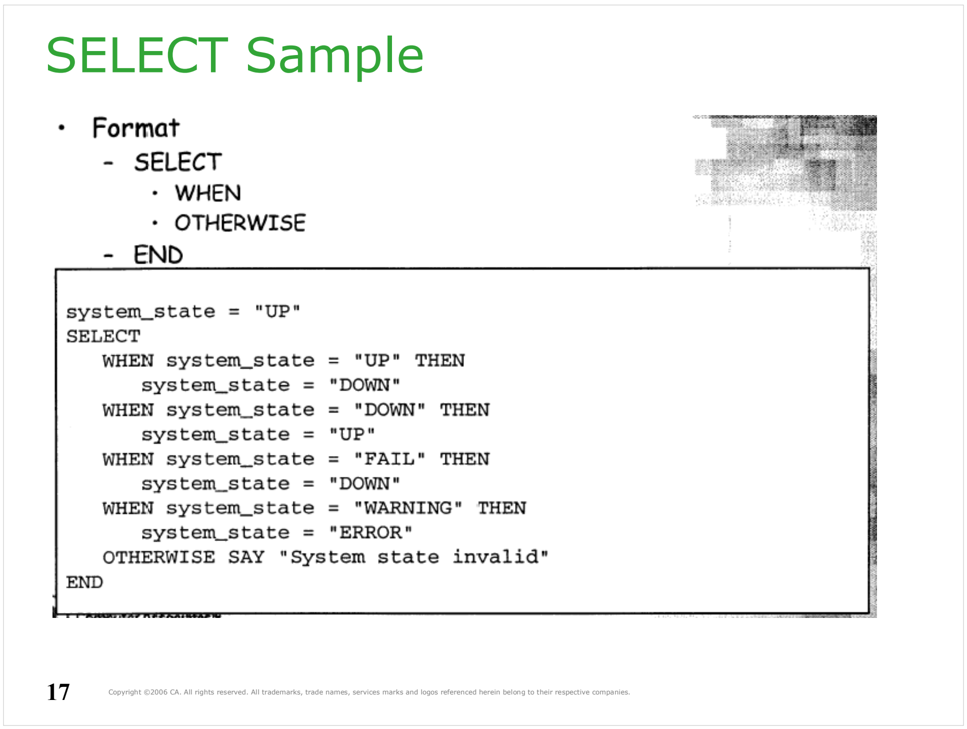
SELECT



SELECT conditionally calls one of several alternative instructions.

Unlike IF with ELSE, the SELECT statement requires the OTHERWISE for all false conditions.

Only the first true choice is evaluated.



SELECT Sample

- Format
  - SELECT
    - WHEN
    - OTHERWISE
  - END

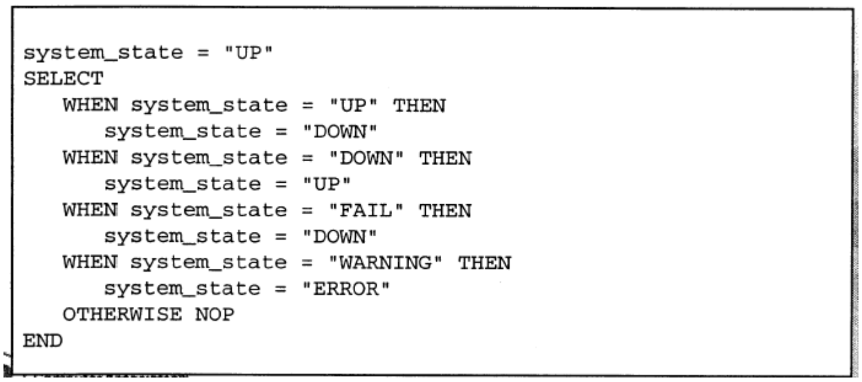
```
system_state = "UP"
SELECT
   WHEN system_state = "UP" THEN
      system_state = "DOWN"
   WHEN system_state = "DOWN" THEN
      system_state = "UP"
   WHEN system_state = "FAIL" THEN
      system_state = "DOWN"
   WHEN system_state = "WARNING" THEN
      system_state = "ERROR"
   OTHERWISE SAY "System state invalid"
END
```

17

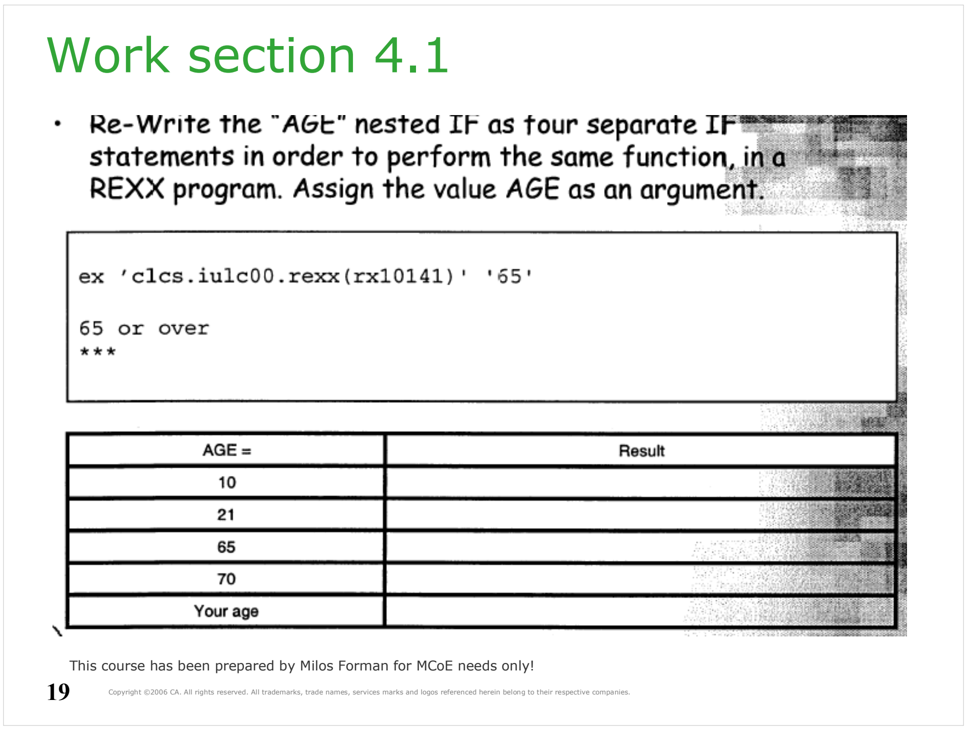
See ‘MCOE.REXA.REXX(RX201413)’

NOP

- Dummy instruction that has no effect
- Often used with and IF and SELECT



```
system_state = "UP"
SELECT
   WHEN system_state = "UP" THEN
      system_state = "DOWN"
   WHEN system_state = "DOWN" THEN
      system_state = "UP"
   WHEN system_state = "FAIL" THEN
      system_state = "DOWN"
   WHEN system_state = "WARNING" THEN
      system_state = "ERROR"
   OTHERWISE NOP
END
```



Work section 4.1

- Re-Write the "AGE" nested IF as four separate IF statements in order to perform the same function, in a REXX program. Assign the value AGE as an argument.

```
ex 'clcs.iulc00.rexx(rx10141)' '65'

65 or over
***
```

AGE =	Result
10	
21	
65	
70	
Your age	

This course has been prepared by Milos Forman for MCoE needs only!

See „Nested IF statement“ on slide 7.

Work section 4.1 (Continued)

```
IF age < 65 THEN
   IF age > 21 THEN
      SAY "Over 21 and under 65"
   ELSE
      IF age >= 16 THEN
         SAY "Between 16 and 21"
      ELSE
          SAY "Under 16"
ELSE
   SAY "65 or over”
```

Enter the results in the table above.

Hint do not use any else statements.

Work section 4.2

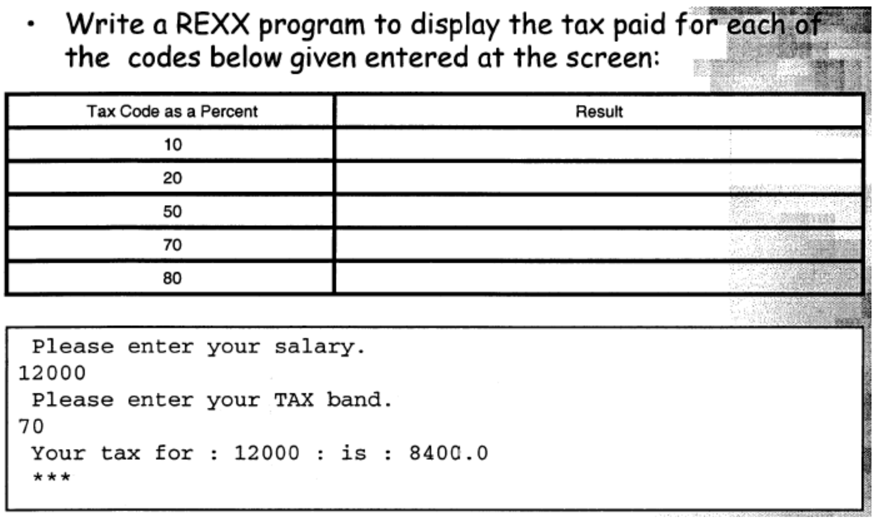
- Re-Write Work section 4.1 "AGE" Using the select Statement

```
ex 'clcs.iulc00.rexx(rx10131)' '65'

65 or over
***
```

AGE =	Result
10	
21	
65	
70	
Your age	

Additional Program



- Write a REXX program to display the tax paid for each of the codes below given entered at the screen:

Tax Code as a Percent	Result
10	
20	
50	
70	
80	

```
 Please enter your salary.
12000
 Please enter your TAX band.
70
 Your tax for : 12000 : is : 8400.0
 ***
```

This course has been prepared by Milos Forman for MCoE needs only!