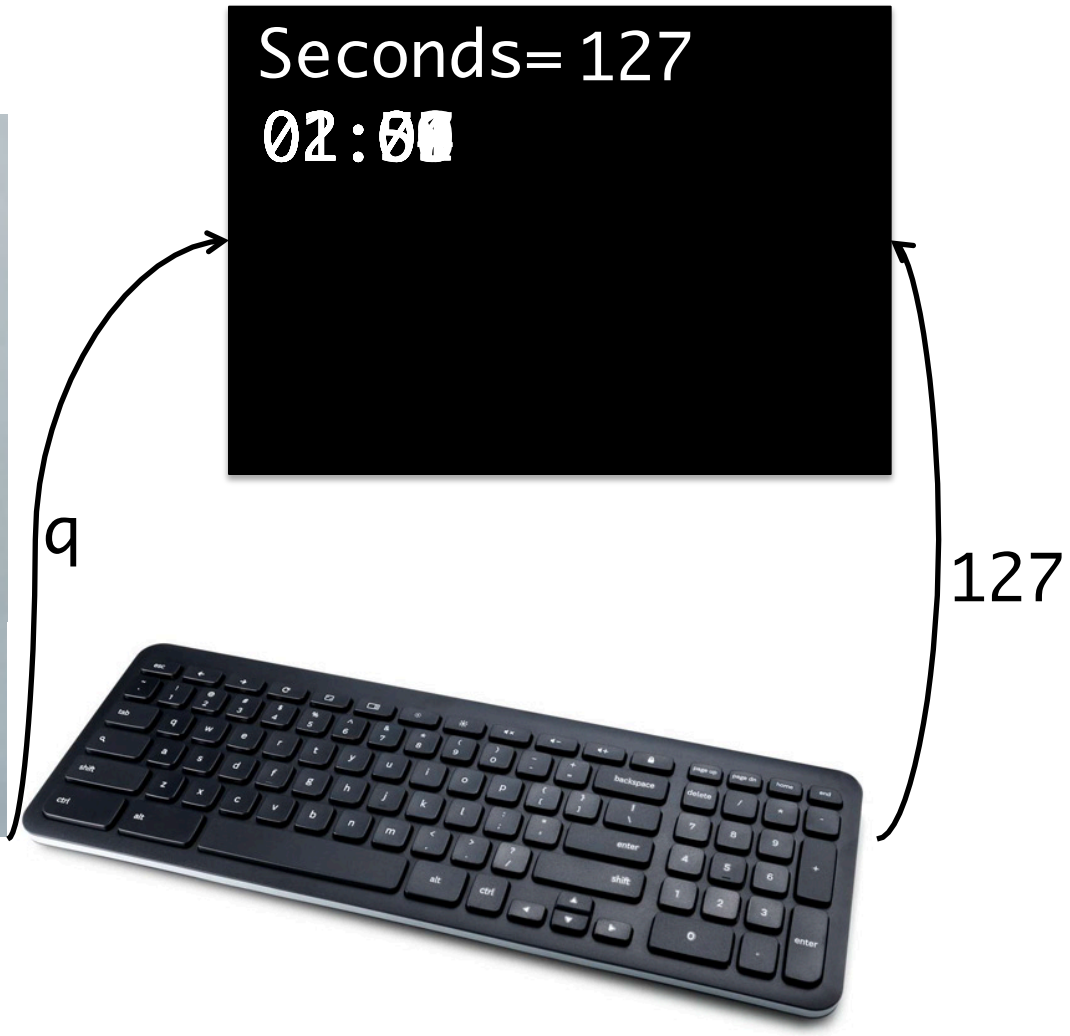
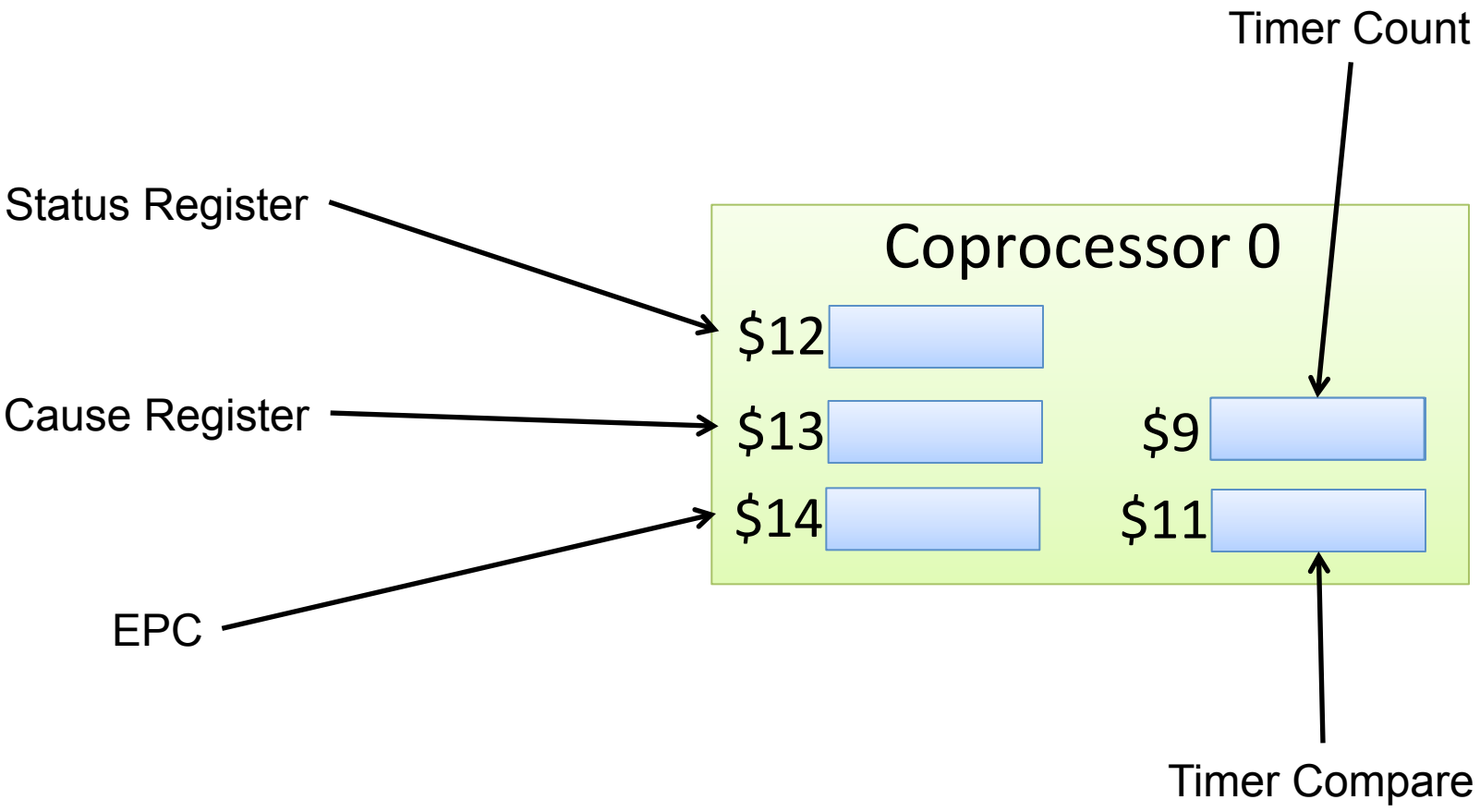
Introduction to Lab # 4: Count-Down Timer

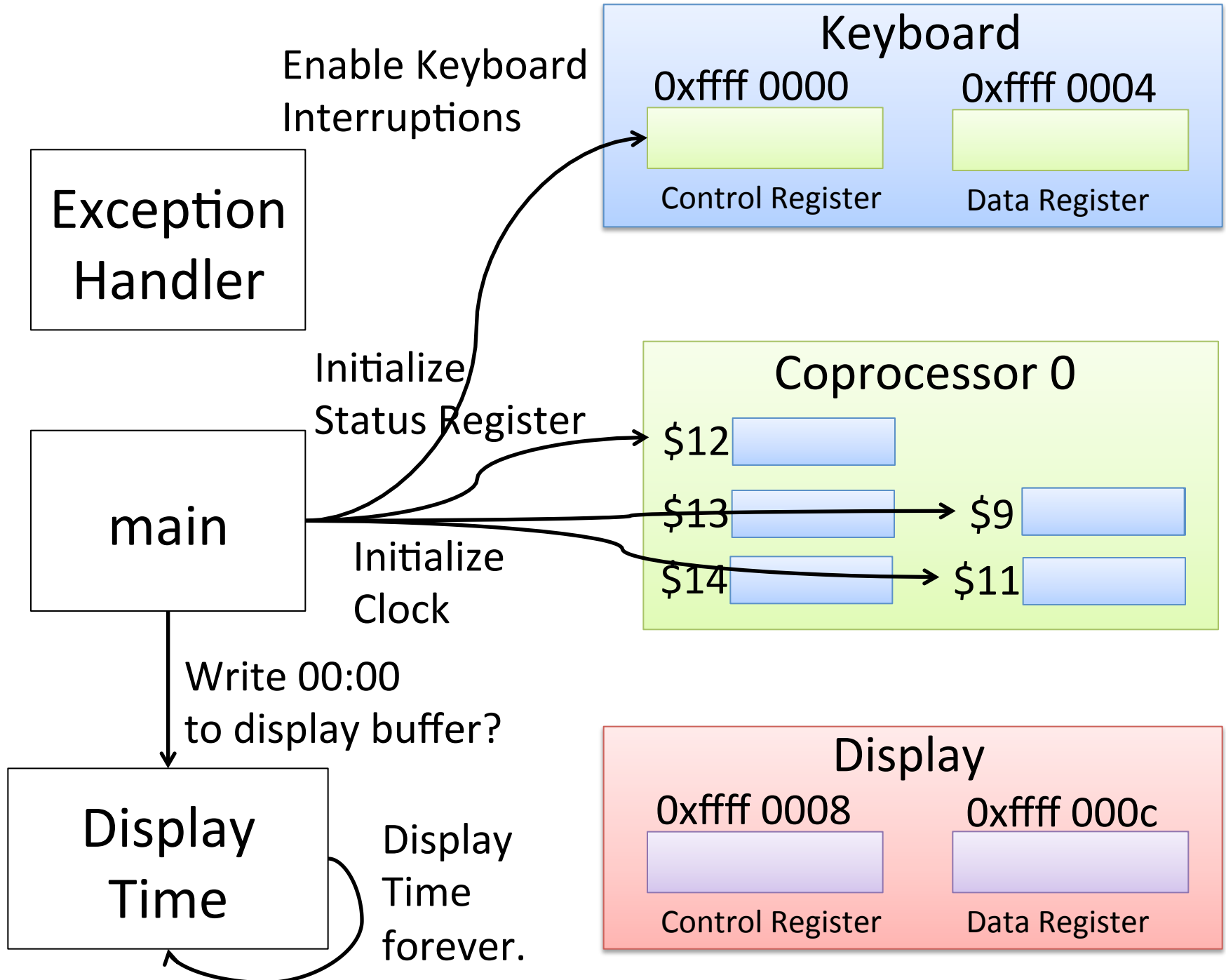
José Nelson Amaral

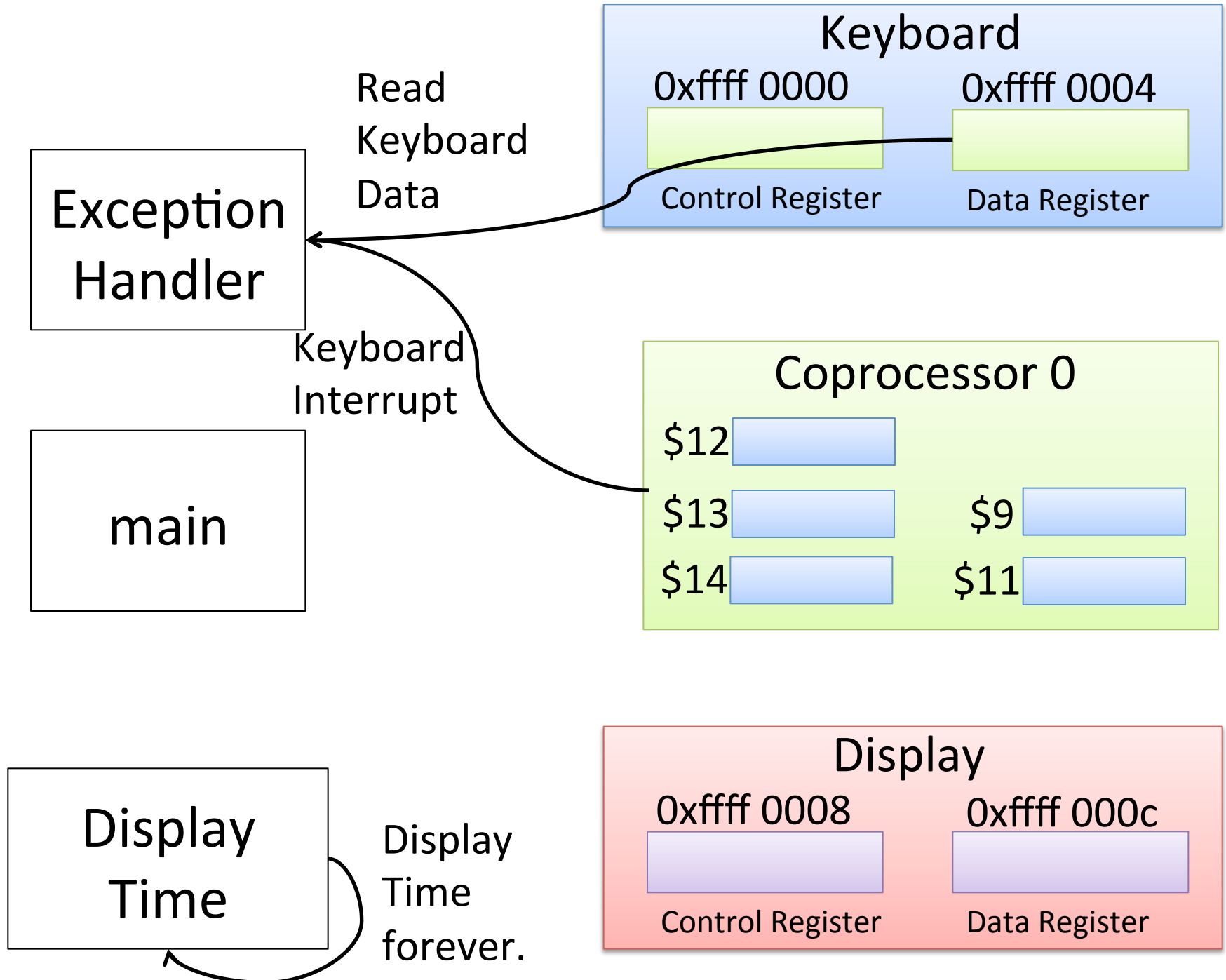
A Digital Count-Down Timer

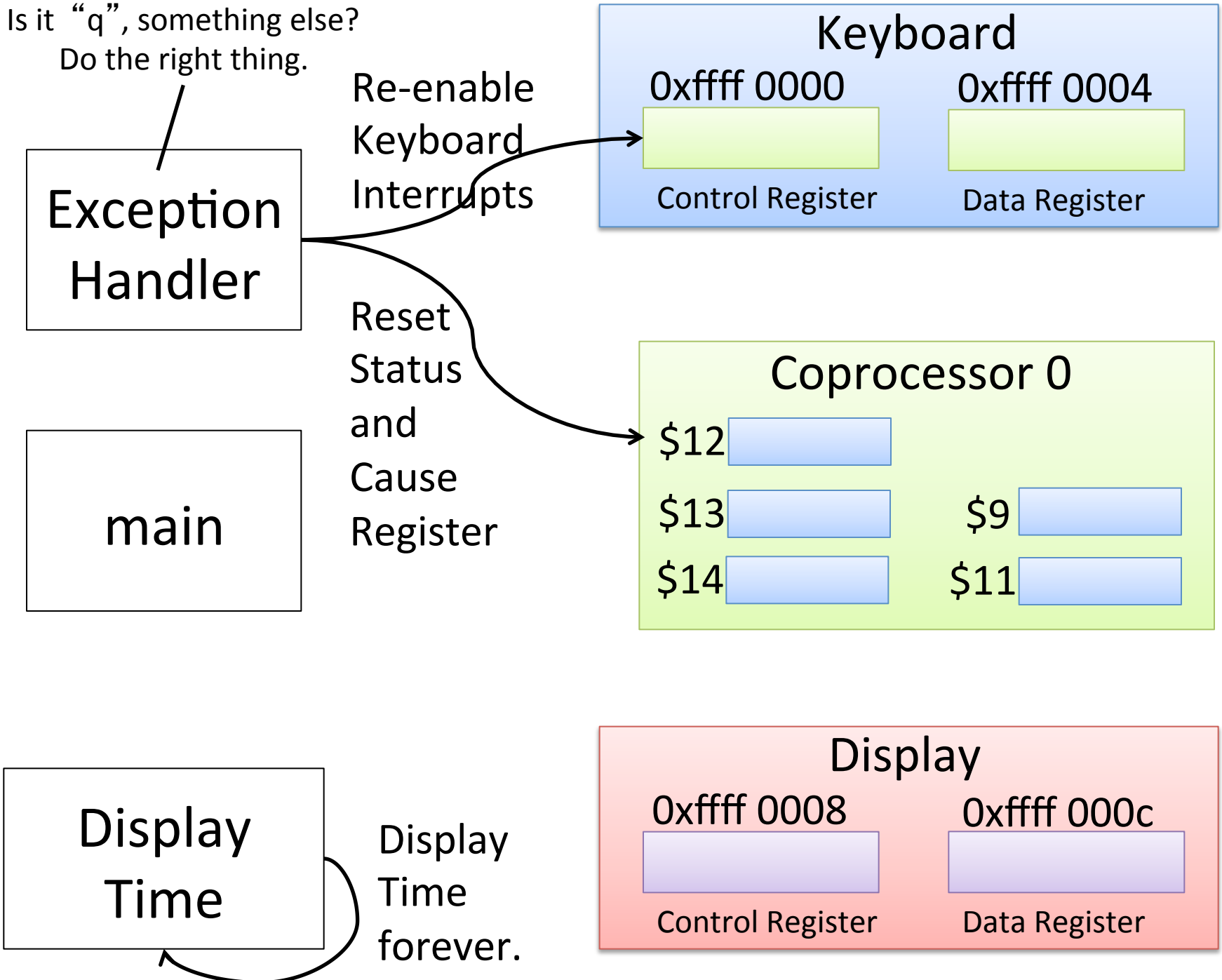


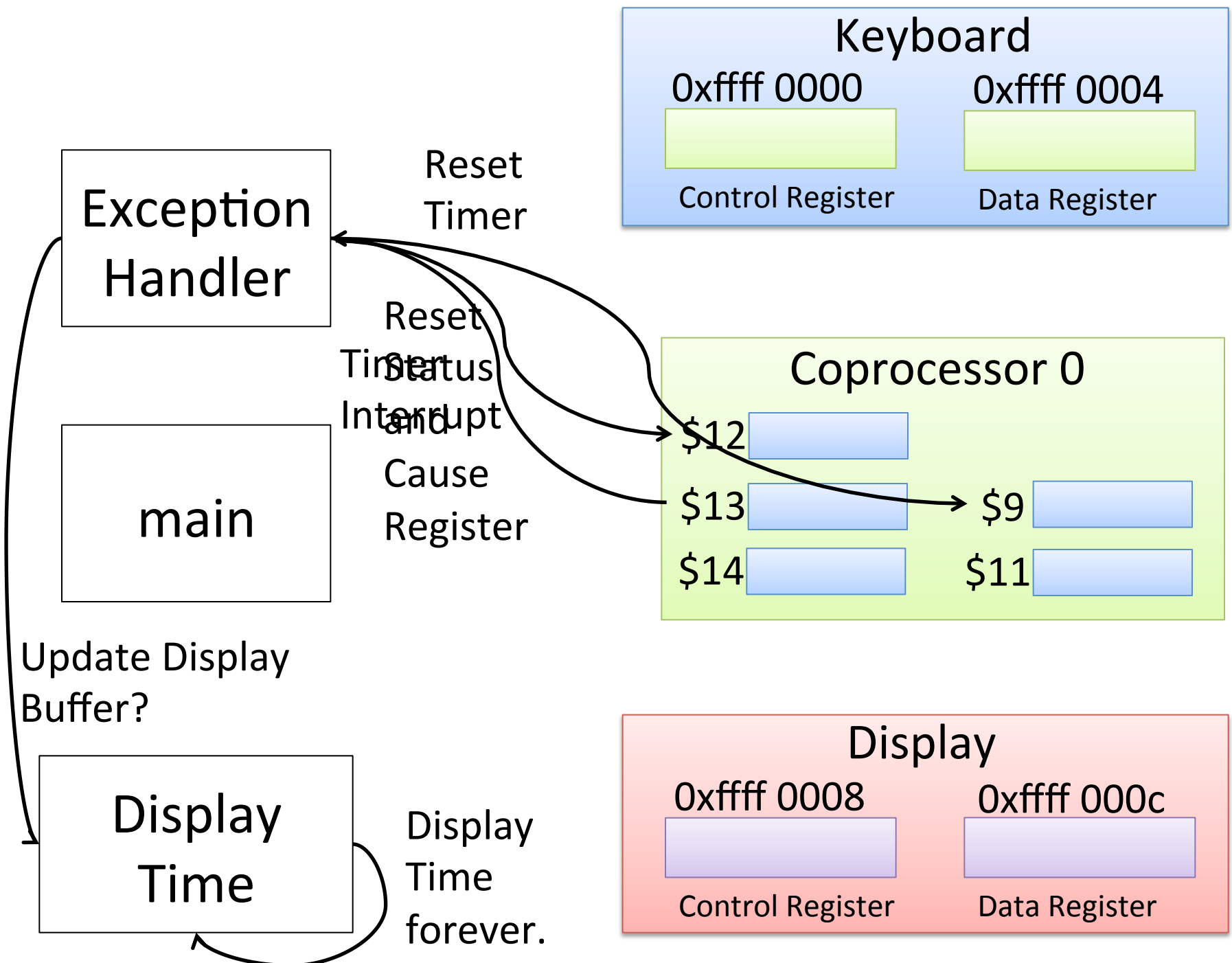
Coprocessor 0



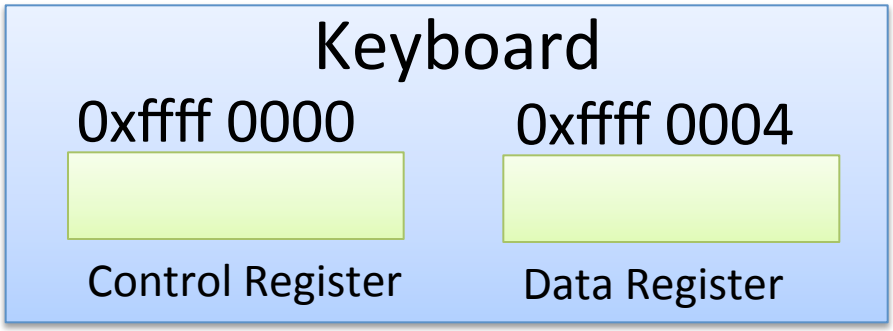

Keyboard
0xffff 0000
0xffff 0004
Control Register
Data Register
Enable Keyboard Interruptions
Exception Handler
Initialize Status Register
Coprocessor 0
$12
$13
$9
$14
$11
main
Initialize Clock
Write 00:00 to display buffer?
Display
0xffff 0008
0xffff 000c
Control Register
Data Register
Display Time
Display Time forever.

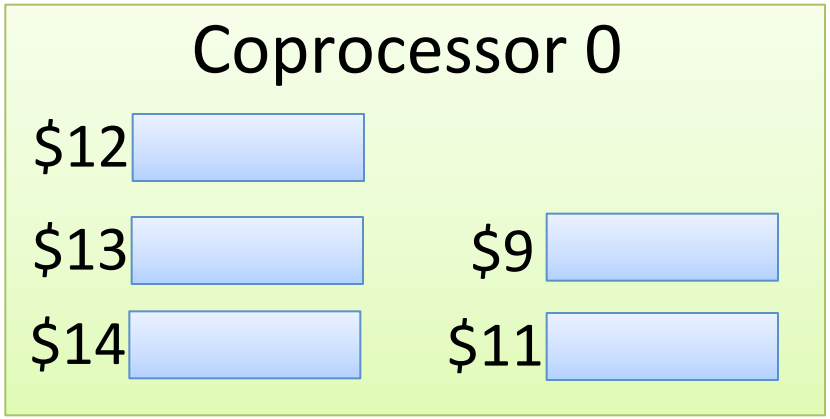

Keyboard
0xffff 0000
0xffff 0004
Control Register
Data Register
Read
Keyboard
Data
Exception
Handler
Keyboard
Interrupt
Coprocessor 0
$12
$13
$14
$9
$11
main
Display
Time
Display
Time
forever.
Display
0xffff 0008
0xffff 000c
Control Register
Data Register


Is it "q", something else?
Do the right thing.
Exception Handler
Re-enable Keyboard Interrupts
Reset Status and Cause Register
main
Keyboard
0xffff 0000
0xffff 0004
Control Register
Data Register
Coprocessor 0
$12
$13
$14
$9
$11
Display Time
Display Time forever.
Display
0xffff 0008
0xffff 000c
Control Register
Data Register


Keyboard
0xffff 0000
0xffff 0004
Control Register
Data Register
Exception Handler
Reset Timer
Reset Timer Interrupt
Status and Cause Register
Coprocessor 0
$12
$13
$14
$9
$11
main
Update Display Buffer?
Display Time
Display Time forever.
Display
0xffff 0008
0xffff 000c
Control Register
Data Register

```
FOREVER:
   p = address of first character
```

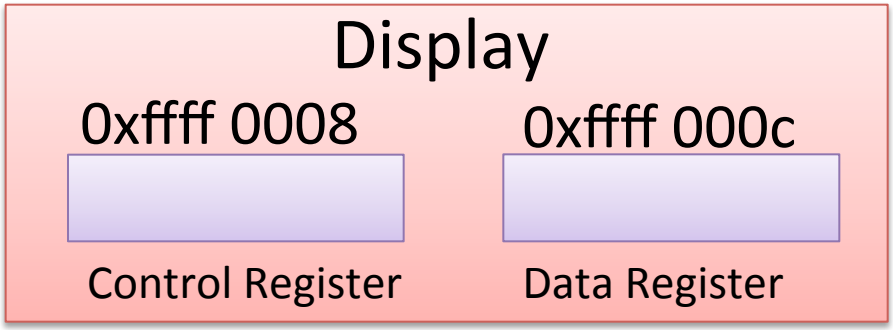




If the routine above is used to display the time, here is an example of the type of string that would be in the display buffer:

Display Buffer:

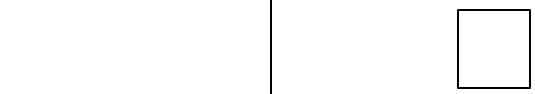
'0'	'0'	':'	'0'	'0'	8	8	8	8	8	0



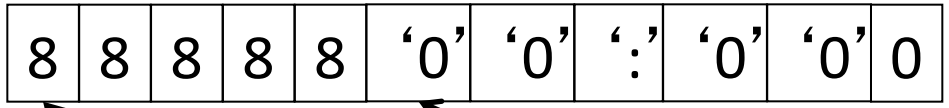
With this solution the clock will be blinking.

```
  q = address of first clock character
NEXT1:
  display_character(q)
  q++
  if *q != NULL
     goto NEXT1
FOREVER:
```

change



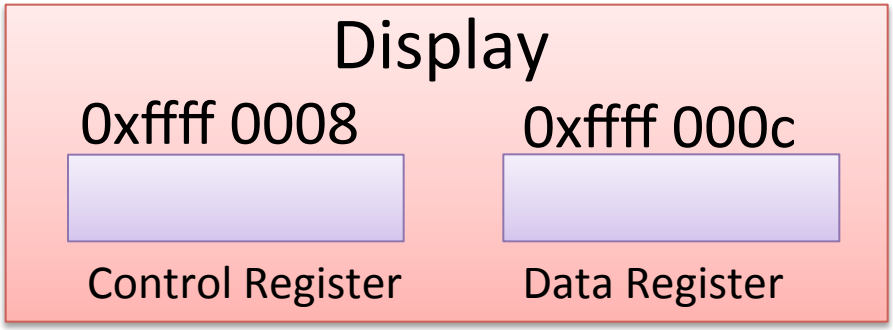
Display Buffer:



First clock character

First buffer character

This is what display_character does.



```
  q = address of first clock character
NEXT1:
  display_character(q)
  q++
  if *q != NULL
     goto NEXT1
FOREVER:
  if change = 0
     goto FOREVER
  change = 0
  p = address of first buffer character
NEXT2:
  display_character(p)
  p++
  if *p != NULL
    goto NEXT2
  go to FOREVER
```

change

Display Buffer:

8	8	8	8	8	'0'	'0'	':'	'0'	'0'	0

These two segments are the same.

```
display_character(c):
POLL:
  read Display Control Register
  if not ready
     go to POLL
  write *c to Display Data Register
return
```

```
  q = address of first clock character
  display_string(q)
FOREVER:
  if change = 0
     goto FOREVER
  change = 0
  p = address of first buffer character
  display_string(p)
  go to FOREVER
```

change

There is only one call to display_character. Thus we may inline it.

Display Buffer:

8	8	8	8	8	‘0’	‘0’	‘:’	‘0’	‘0’	0

```
display_string(s):
NEXT1:
  display_character(s)
  s++
  if *s != NULL
     goto NEXT1
return
```

```
display_character(c):
POLL:
  read Display Control Register
  if not ready
     go to POLL
  write *c to Display Data Register
return
```

```
  q = address of first clock character
  display_string(q)
FOREVER:
  if change = 0
     goto FOREVER
  change = 0
  p = address of first buffer character
  display_string(p)
  go to FOREVER
```

This solution does not work well with xterm (the backspace character prints as ^H). Try to use a different terminal such as xfce4-terminal.

change

There is only one call to display_character. Thus we may inline it.

Display Buffer:

8	8	8	8	8	'0'	'0'	':'	'0'	'0'	0

```
display_string(s):
POLL:
  read Display Control Register
  if not ready
     go to POLL
  write *s to Display Data Register
  s++
  if *s != NULL
     goto POLL
return
```

Need to fix the exception handler so that it updates "change" when there is a new time to display.