Flashing an “LED” (Part 1)
Busy wait timer

Computer Systems Lecture 8.4

Let’s simulate a flashing LED

- We can simulate a flashing LED by drawing a single pixel in ARMLite’s graphical display
- ARMlite has three modes of pixel graphics display:
  - Low resolution (32 x 24 pixels)
  - Medium resolution (64 x 48 pixels)
  - High resolution (128 x 96 pixels)
- For this we will use the low resolution modality

ARMlite Low Res Pixel Graphics

- In ARMLite low-res modality, there are 32 x 24 = 768 pixels.
- The display window is mapped to a contiguous block of memory.
- Pixel colours are specified as 24 bit Red-Green-Blue values (8 bits per colour component)
  - Eg Bright Green is 0x000FF000, White is 0x00FFFFFF
  - In ARMlite only the least significant 24 bits of each word are used
- Each pixel therefore has an address, and can be written to (STR) or read from (LDR) like any memory location

ARMlite Low Res Pixel Graphics

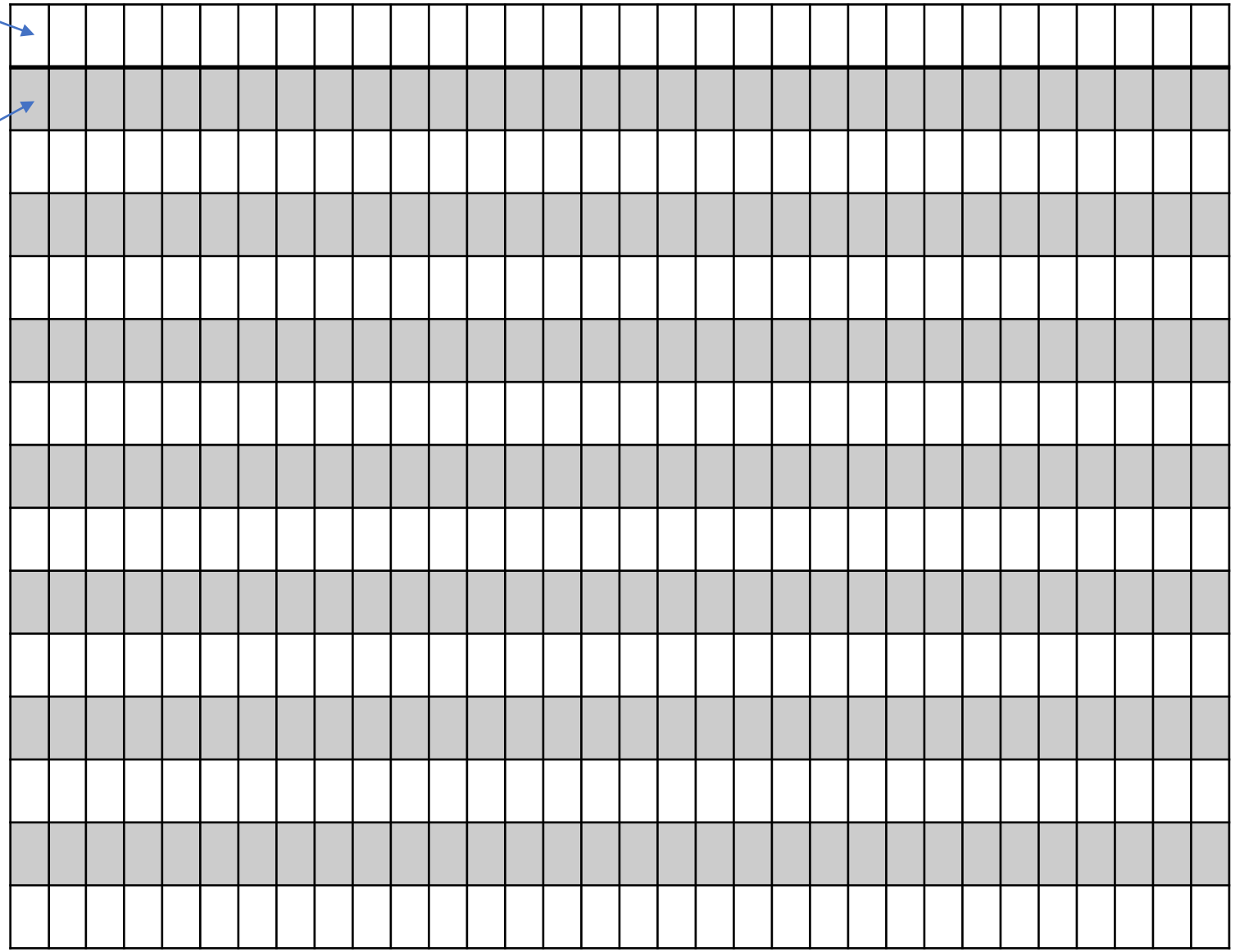
- In low-res mode, each pixel is assigned a label, starting from top left and moving across each row:
  - .Pixel0, .Pixel1, .Pixel2 …. .Pixel767
- For convenience, labels pre-exist for standard colour codes:
  - Eg., the colour green can be represented by the predefined label .green
  - .green == #0x00008000

ARMlite Low Res Pixel Graphics

.Pixel0

.Pixel31

.Pixel32

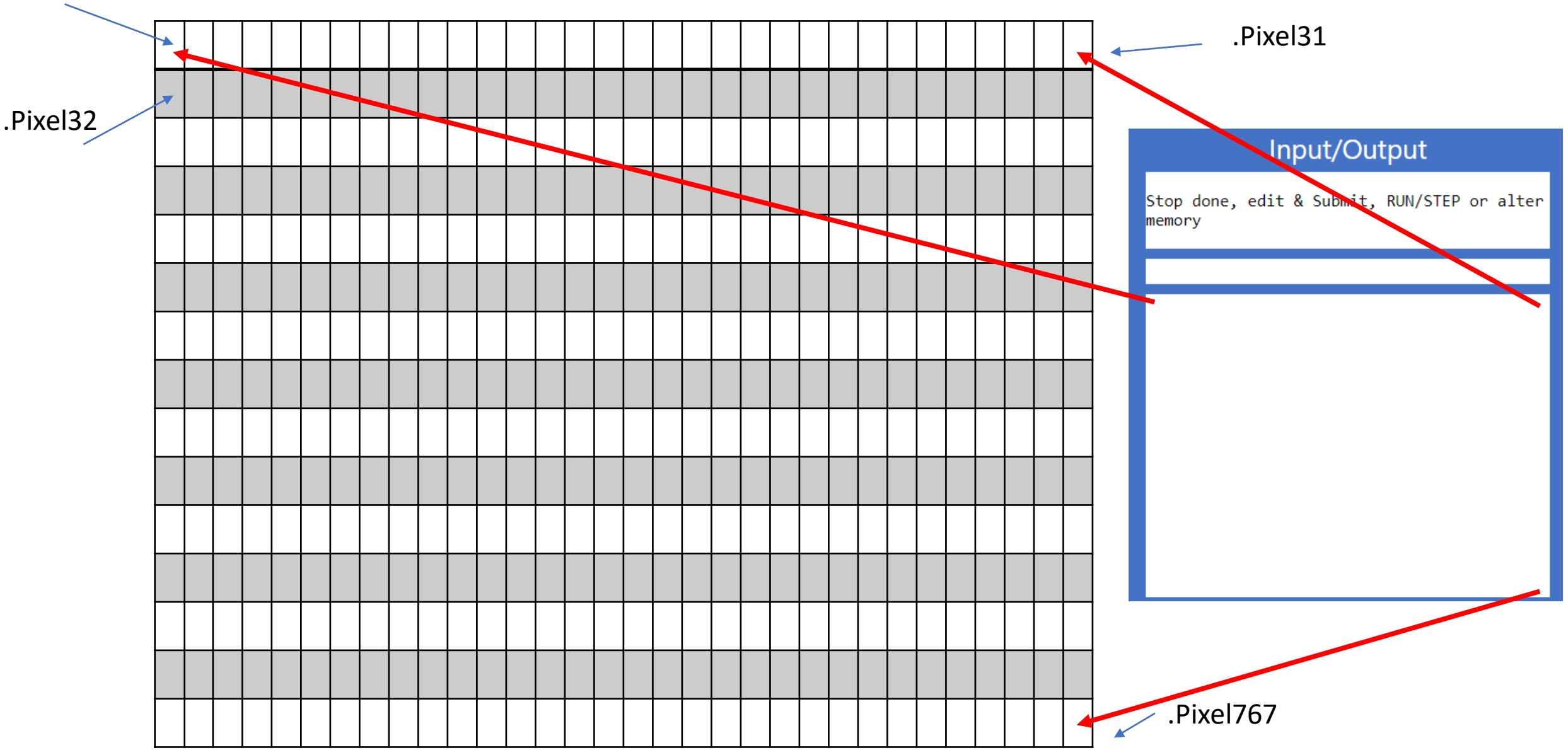


.Pixel767

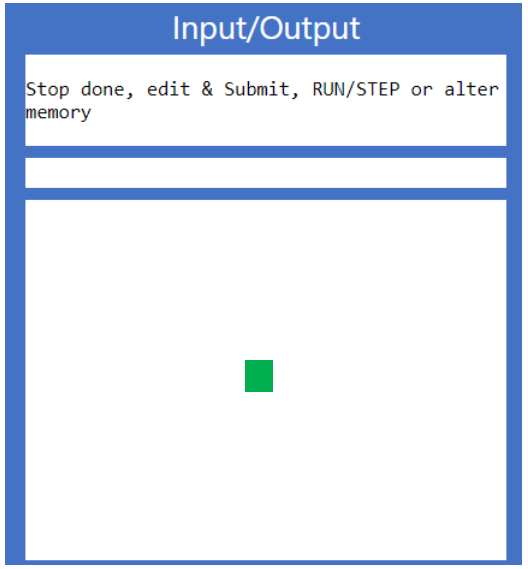
Input/Output

Stop done, edit & Submit, RUN/STEP or alter memory

ARMlite Low Res Pixel Graphics



ARMlite Low Res Pixel Graphics



Let’s paint a green pixel in the middle of the display:

The index of the middle of the display will be 11*32 + 15 = 367

```
MOV R0, #.green      // set R0 to RGB for green
STR  R0, .Pixel367   // write R0 to address of Pixel 367
```

OK – Now let's simulate a flashing LED

- To simulate a flashing green LED, we will need to paint a pixel green and then white. Maybe this ?

```
MOV R0, #.green
MOV R1, #.white
flash:
STR R0, .Pixel367
STR R1, .Pixel367
B flash
```

OK – Now let's simulate a flashing LED

- To simulate a flashing green LED, we will need to paint a pixel green and then white. Maybe this ?

```
MOV R0, #.green
MOV R1, #.white
flash:
STR R0, .Pixel367
STR R1, .Pixel367
B flash
```

Pause video and run this in ARMlite – any issues with this ?

OK – Now let's simulate a flashing LED

- Issue - the flashing is too fast !
- In fact, in a real system this would probably not produce a flash at all, just a dimmer LED

```
MOV R0, #.green
MOV R1, #.white
flash:
STR R0, .Pixel367
STR R1, .Pixel367
B flash
```

OK – Now let's simulate a flashing LED

- Issue - the flashing is too fast !
- In fact, in a real system this would probably not produce a flash at all, just a dimmer LED

```
MOV R0, #.green
MOV R1, #.white
flash:
STR R0, .Pixel367
STR R1, .Pixel367
B flash
```

The issue is here – we need to insert a delay between the "ON" and the "OFF"

OK – Now let's simulate a flashing LED

- Issue - the flashing is too fast !
- In fact, in a real system this would probably not produce a flash at all, just a dimmer LED

```
MOV R0, #.green
MOV R1, #.white
flash:
STR R0, .Pixel367
STR R1, .Pixel367
B flash
```

The issue is here – we need to insert a delay between the "ON" and the "OFF"

For this – we need a timer!

A “Busy Wait” Timer

- Pseudocode:
  - Set up pixels for display
  - Loop1:
    - Turn LED on (paint green pixel)
    - Busy wait
    - Turn LED off (paint white pixel)
    - Busy wait
  - branch to loop1

A “Dumb” Busy Wait Timer

- Busy wait:
  - a loop that executes until it reaches a certain value
  - keep CPU occupied

```
set limit value    // some large number
initialise counter to 0
timer:
   • Add 1 to counter
   • Compare counter to limit value
   • branch to timer if  counter < limit
```

A “Dumb” Busy Wait Timer

- Busy wait:
  - a loop that executes until it reaches a certain value
  - keep CPU occupied

```
set limit value    // some large number
initialise counter to 0
timer:
   • Add 1 to counter
   • Compare counter to limit value
   • branch to timer if  counter < limit
```

So we need a conditional branch as part of the timer loop !

A “Dumb” Busy Wait Timer

```
MOV R2, #5000000          // init number of iterations for timer
MOV R2,R3                 // MOV iterations into working register R3
timer1:
      SUB R3,R3,#1        // subtract 1 from R3
      CMP R3, #0          // compare R3 with #0
      BNE timer1          // keep looping until R3 reaches zero
```

Incorporating timer into LED flash

- **Recall pseudo code:**

```
Set up pixels for display
Loop1:
   Turn LED on (paint green pixel)
   Busy wait
   Turn LED off (paint white pixel)
   Busy wait
branch to loop1
```

Incorporating timer into LED flash

```
 1|       mov r0, #.green
 2|       mov r1, #.white
 3|       mov r2, #5000000
 4|flash:
 5|       str r0,.Pixel367  ; flash on
 6|       mov r3,r2
 7|timer1:
 8|       sub r3,r3,#1
 9|       CMP r3,#0
10|       BNE timer1
11|       str r1,.Pixel367  ; flash off
12|       mov r3,r2
13|timer2:
14|       sub r3,r3,#1
15|       CMP r3,#0
16|       BNE timer2
17|       B flash
18|       halt
```