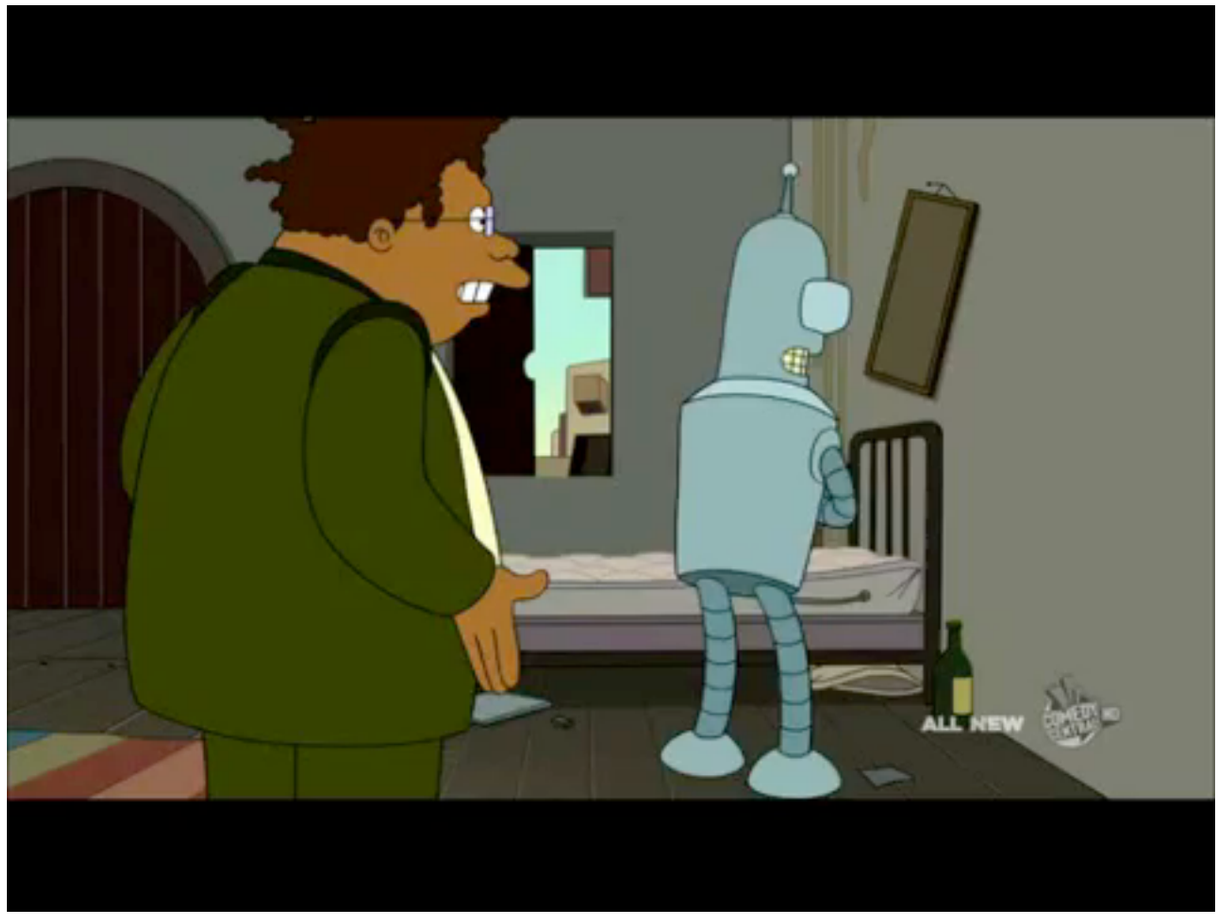
GPIO II & Assembly V

ECE 3710

how goes lab?



conflicting emotions?
want to hit things? ⟶ you're doing it right, then

Don't sweat the petty things and don't pet the sweaty things.

- George Carlin

example: alarm system monitor

PD.3

if sensor outputs 1, do something:

PD.0=1

1. trigger alarm
2. lock doors
3. execute notification routine

PD.1--2=1

example: alarm system monitor

```
; 3. set port pins as INPUT(0)/OUTPUT(1)
LDR R1,=PD_DIR_R
MOV R0,#0x7 ;0x7=0b0111 (P0--2=output;P3=input)
STR R0,[R1]

; let's be ready for alarm
MOV R2,#0xF ;write this to port to set off alarm and lock doors
; data for port d: everything should be zero initially
LDR R1,=PD_DATA_R
MOV R0,#0x0
STR R0,[R1]
```

note: default input config is basically pull-up;
so input must be changed in GPIO box

```
chksnsr
  LDR R0,[R1] ;check sensor by reading PD.3
  AND R0,#0x8 ;0x8=0b1000
  CMP R0,#0x8 ;if PD.3==1 then intruder
  BEQ intruder
  B chksnsr

intruder
  STR R2,[R1] ;sound alarm and lock doors
  BL notify
```

notice:

masking pins is tedious → is there a better way?

```
;disable i/o for pins zero and seven on port b
;(don't disturb previous settings)
ldr R1,=PB_DEN_R ;get addr. enable reg
ldr R0,[R1] ;get i/o config for port
and R0,#0x7E ;0x7E = 0b01111110
str R0,[R1]
```

why do I ask?

the student:
exhausted and sweaty from time
spent in the lab and on homework

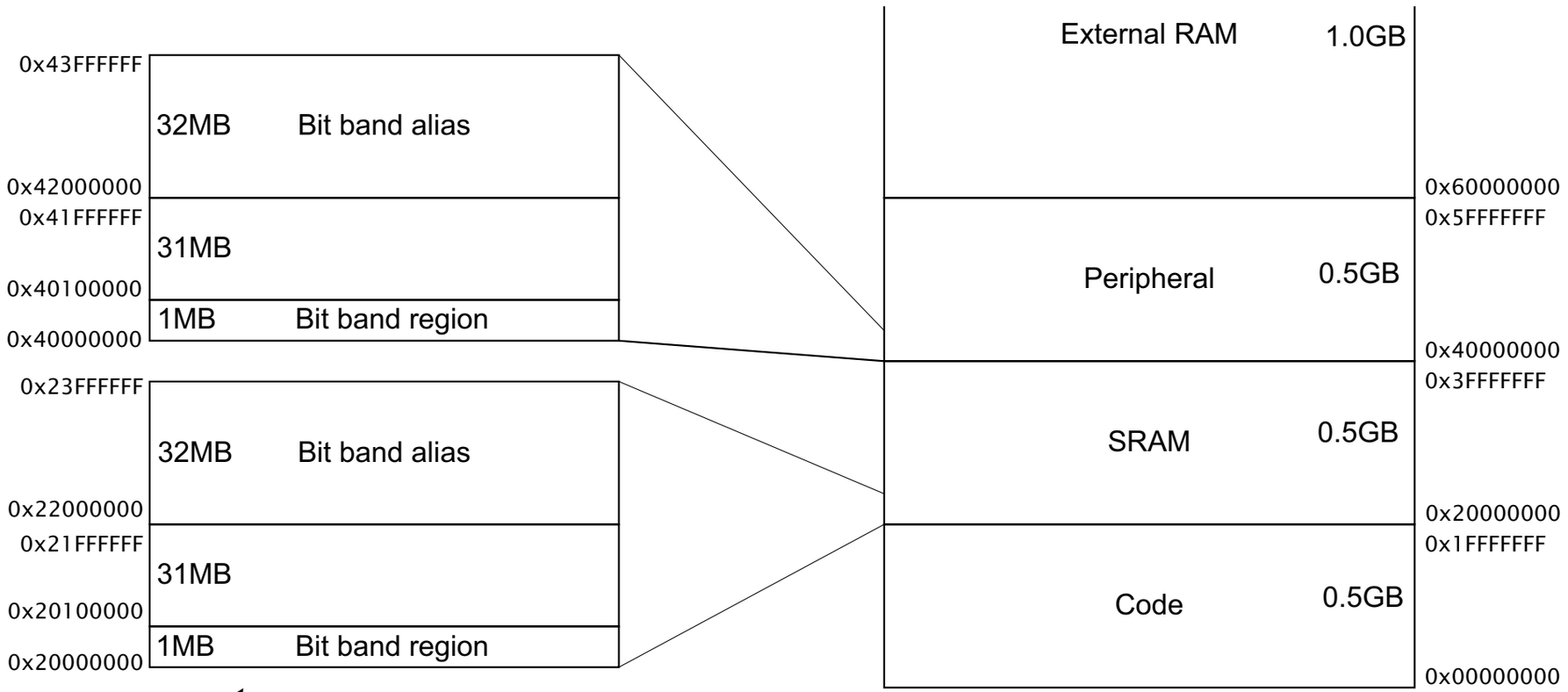
1. another concept
(get your money worth)
2. make things easier
(long run)

bit banding: addressing (R/W) individual bits

2. these addresses

(i.e. these addresses refer to those bits)

using



1. we R/W to individual bits here

why 32 MB for alias?

0x20100000-0x20000000

=0x100000

=0b10000000000000000000

= 2^20 bytes (1 MB)

= 8*2^20 bits (8 MB to addr them)

seem to need only

aliased addr. is word aligned

(end in 0,4,8,C)

each bit
actually gets 4

= 4*8*2^20 bits (32 MB)

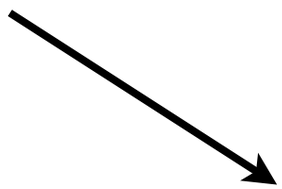
take-away:each word in alias refers to a bit in bit band

bit band syntax

0x22000000+32*n*+4**b*

the bit address for the
*b*th bit of byte address 0x20000000+*n*

e.g. third bit of 0x20000123



0x22000000+32*0x*123*+4**3*
=0x22000000+*0x2460*+*12*
=0x2200246C

bit ordering: 31--0

example: write 0xF to 0x20000123

(byte address)

hard way:

```
mov R0,#0x1
;0x22000000+32*0x123+4*0
;=0x22002460
ldr R1,=0x22002460
str R0,[R1]
add R1,#4
str R0,[R1]
add R1,#4
str R0,[R1]
add R1,#4
str R0,[R1]
```

note: `str` moves LSB of register;
`ldr` sets LSB of register

???

easy way:

```
mov R0,#0xF
ldr R1,=0x20000123
str R0,[R1]
```

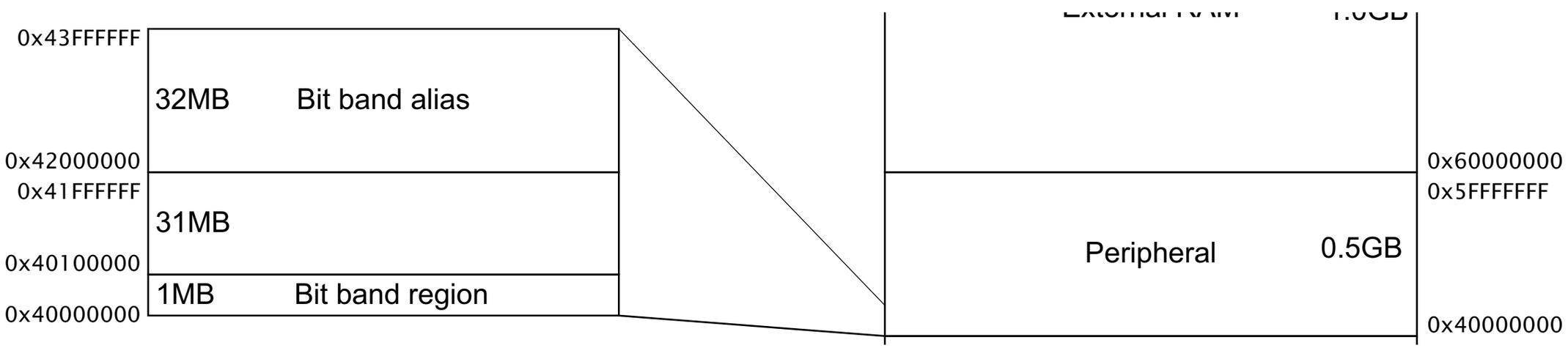
potentially overwrite
0x124,5,6

example: single pin on port

0x42000000+32*n*+4*b*

the bit address for the *b*th bit of byte address 0x40000000+*n*

bit band for I/O:



example: copy pin one on port d to R0

```
      ;0x400073FC => 0x42000000 + 32*0x73FC + 4*1 = 0x420E7F84
PD1_DATA_B EQU 0x420E7F84
       ;0x40007400 => 0x42000000 + 32*0x7400 + 4*1 = 0x420E8004
PD1_DIR_B EQU 0x420E8004
       ;0x40007420 => 0x42000000 + 32*0x7420 + 4*1 = 0x420E8404
PD1_AF_B EQU 0x420E8404
       ;0x4000751C => 0x42000000 + 32*0x751C + 4*1 = 0x420EA384
PD1_EN_B EQU 0x420EA384
```

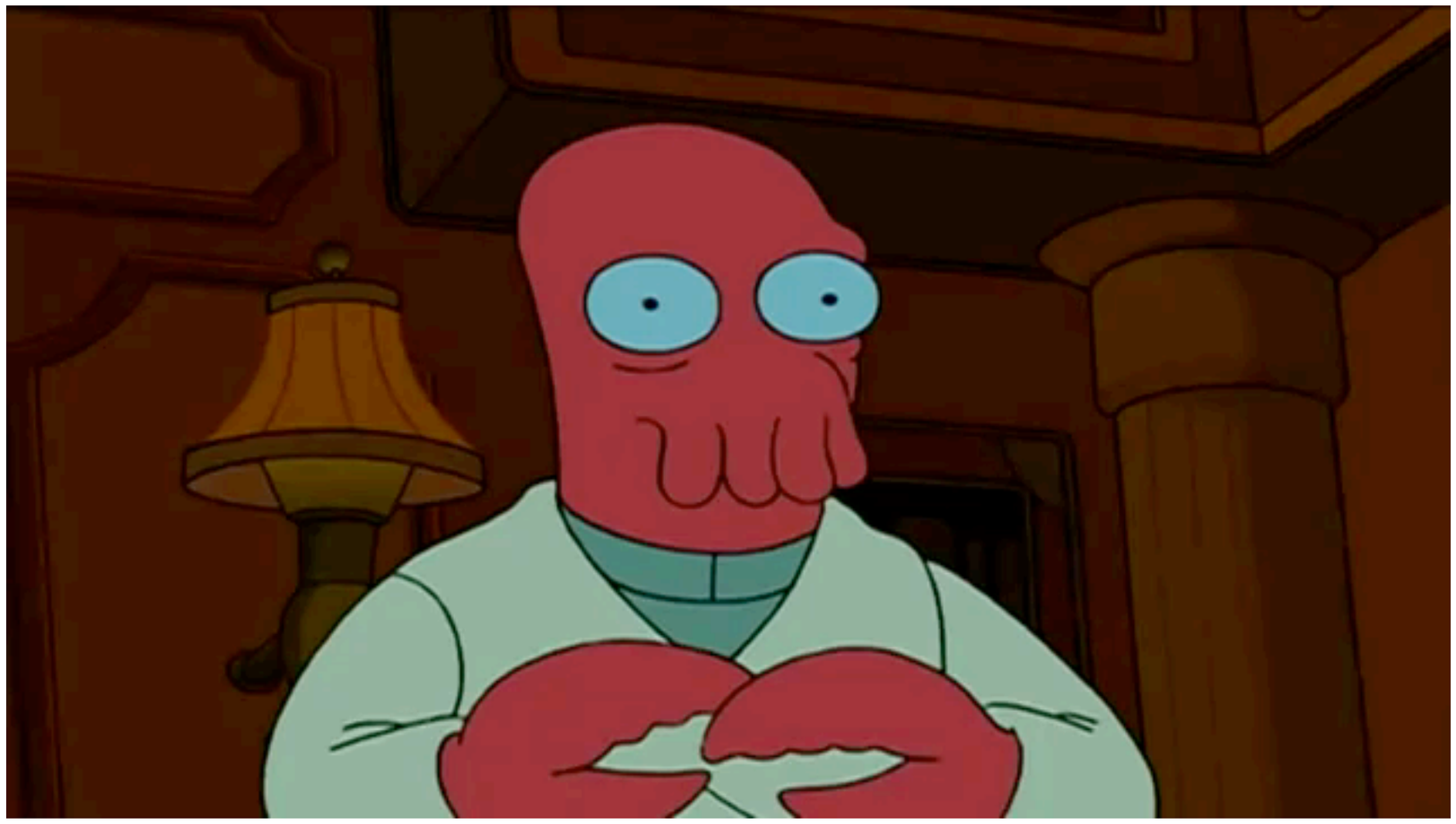
all refer to the one bit

$4000.73FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTD_DATA_R
$4000.7400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTD_DIR_R
$4000.7420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTD_AFSEL_R
$4000.751C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTD_DEN_R

```
      ;0x400FE108 => 0x42000000 + 32*0xFE108 + 4*3 = 0x43FC210C
CLK_PD_B EQU 0x43FC210C
```

$400F.E108	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGC2_R

hrm...more work than intended...



you'll get over it, eh?

(easy to define and reuse)