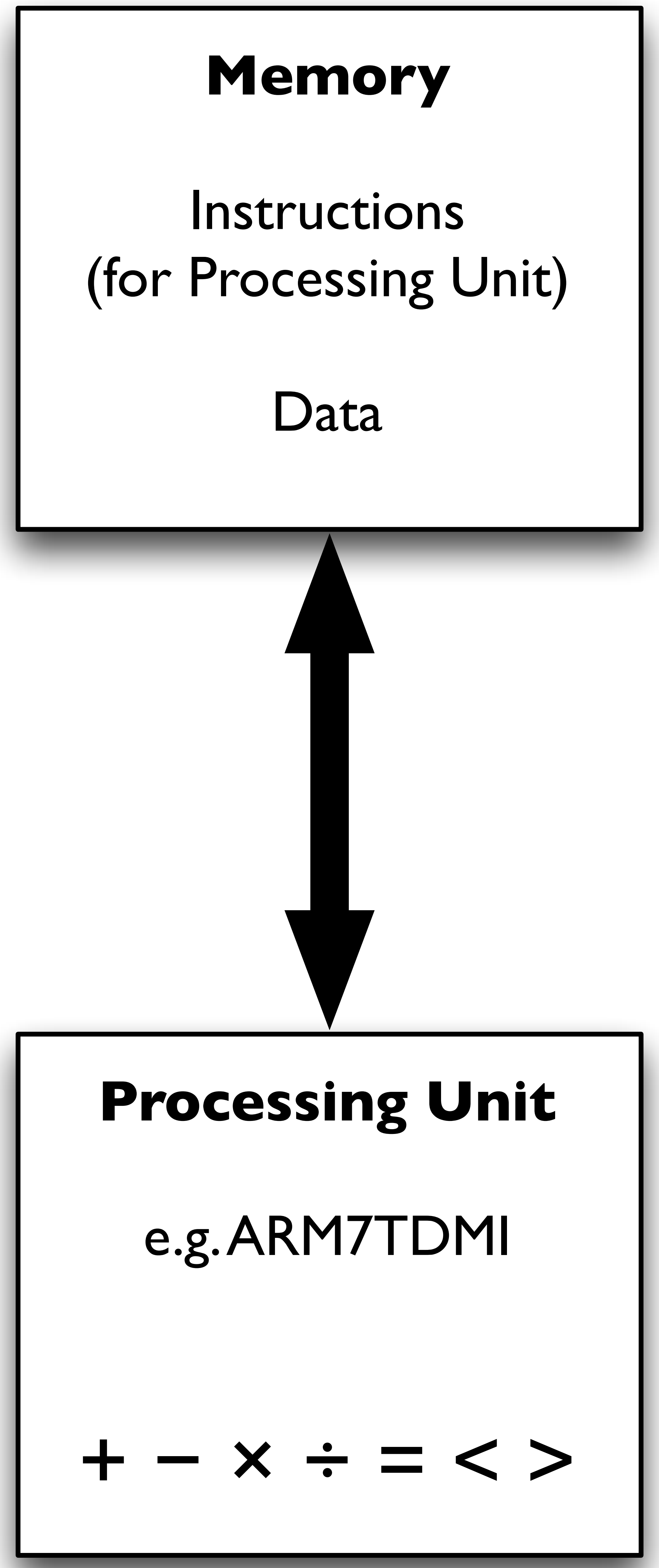
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

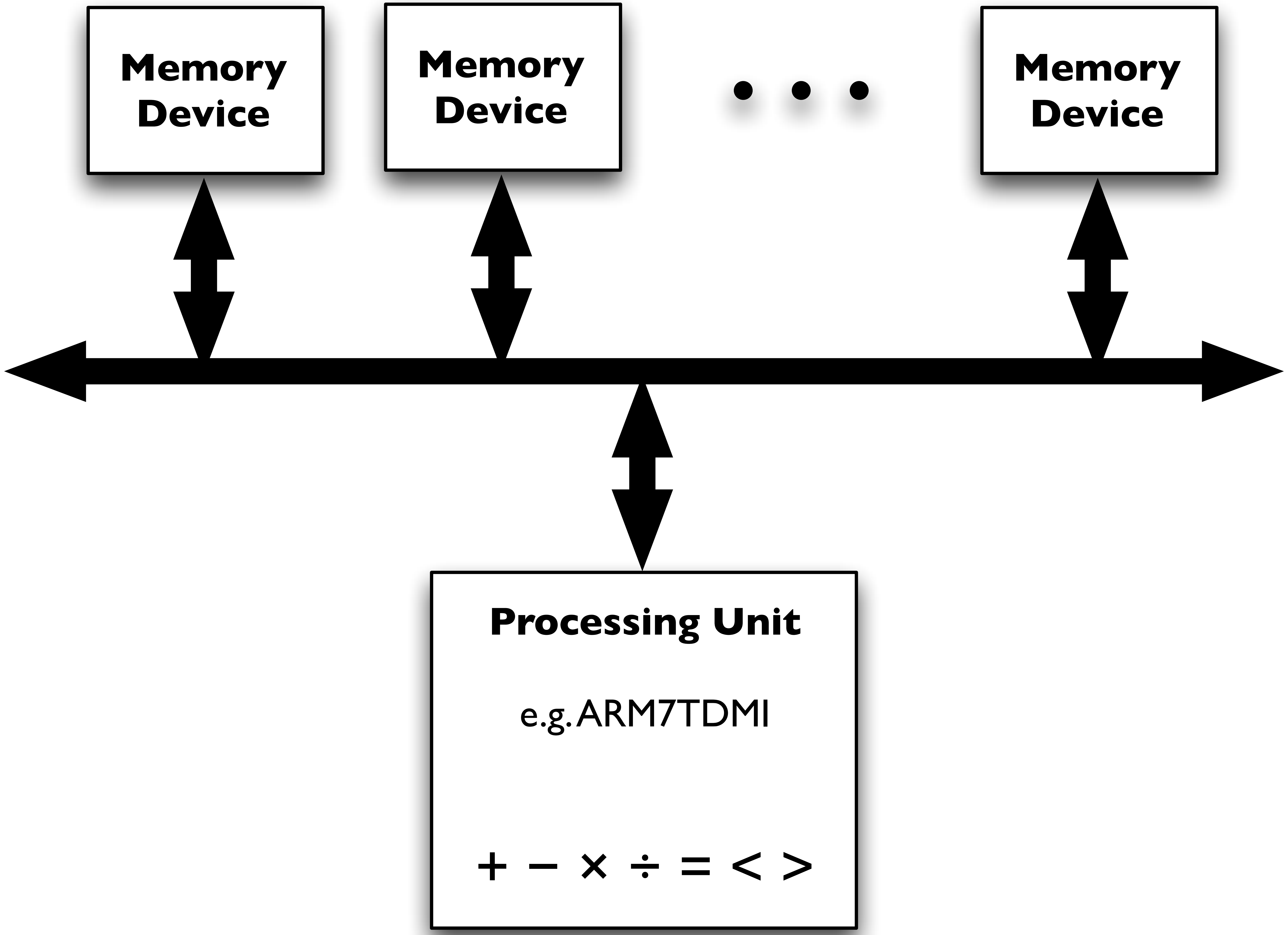
05 – Memory-Mapped I/O

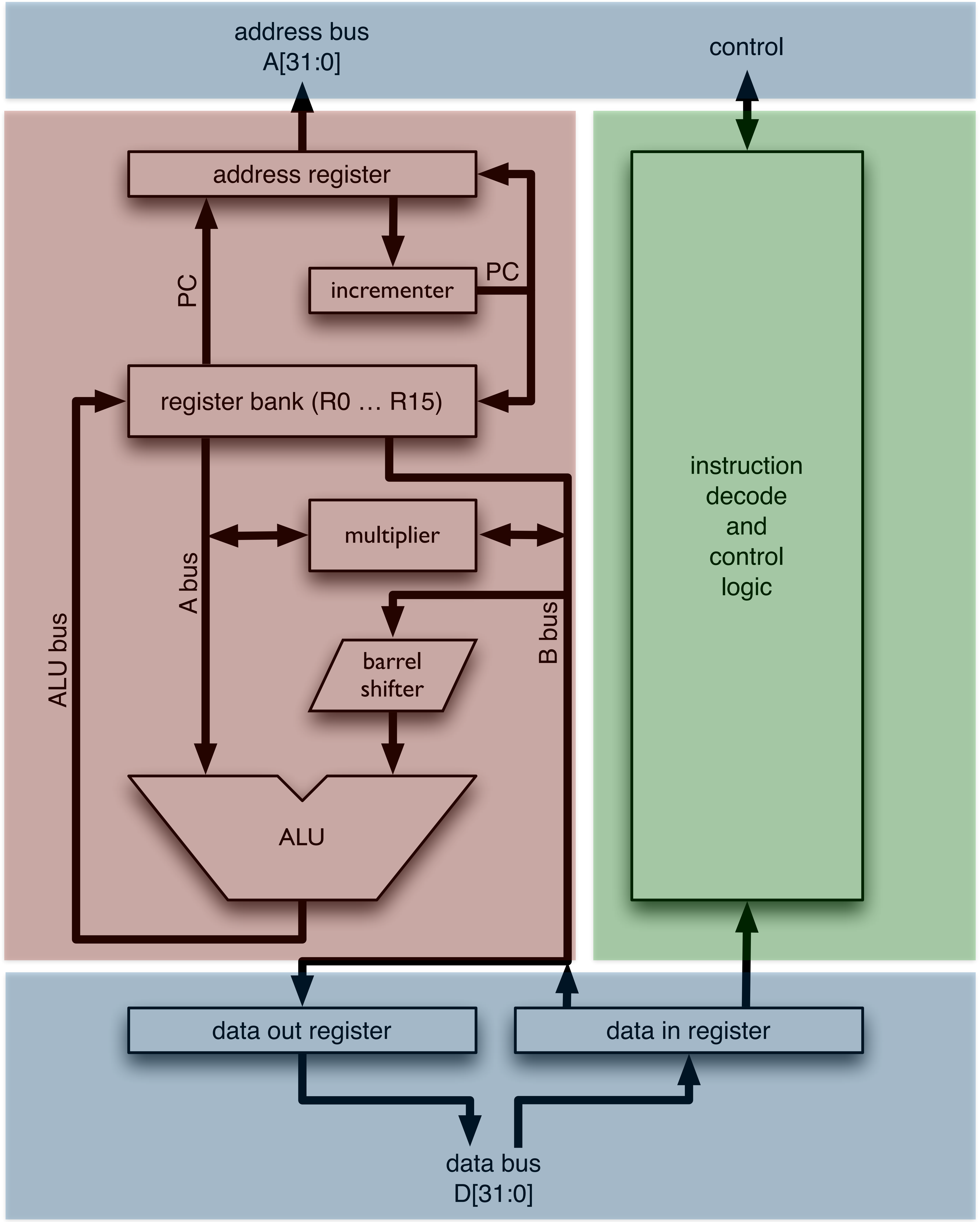
CS1022 – Introduction to Computing II

Dr Jonathan Dukes / jdukes@scss.tcd.ie
School of Computer Science and Statistics

Simple Model of a Microprocessor

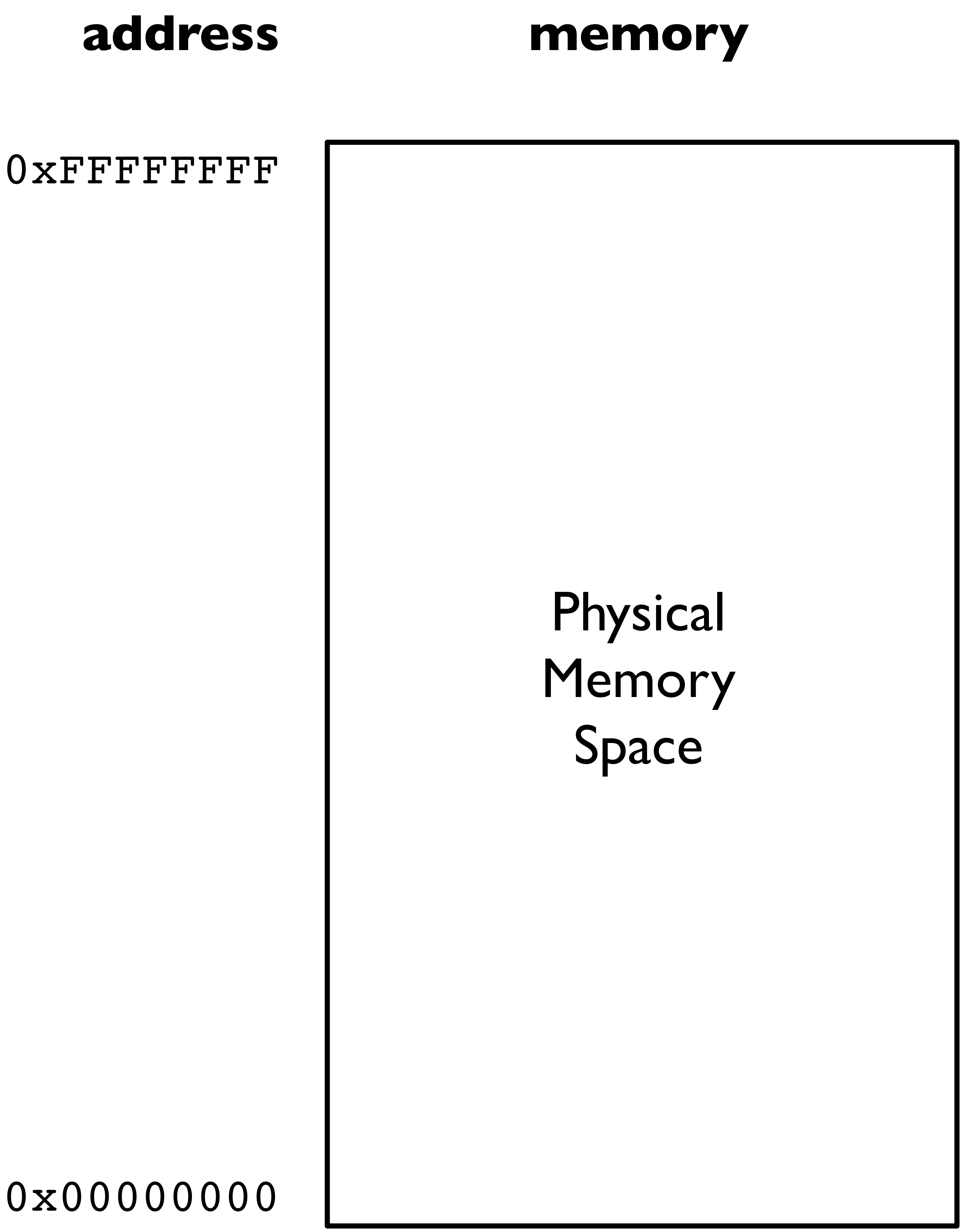


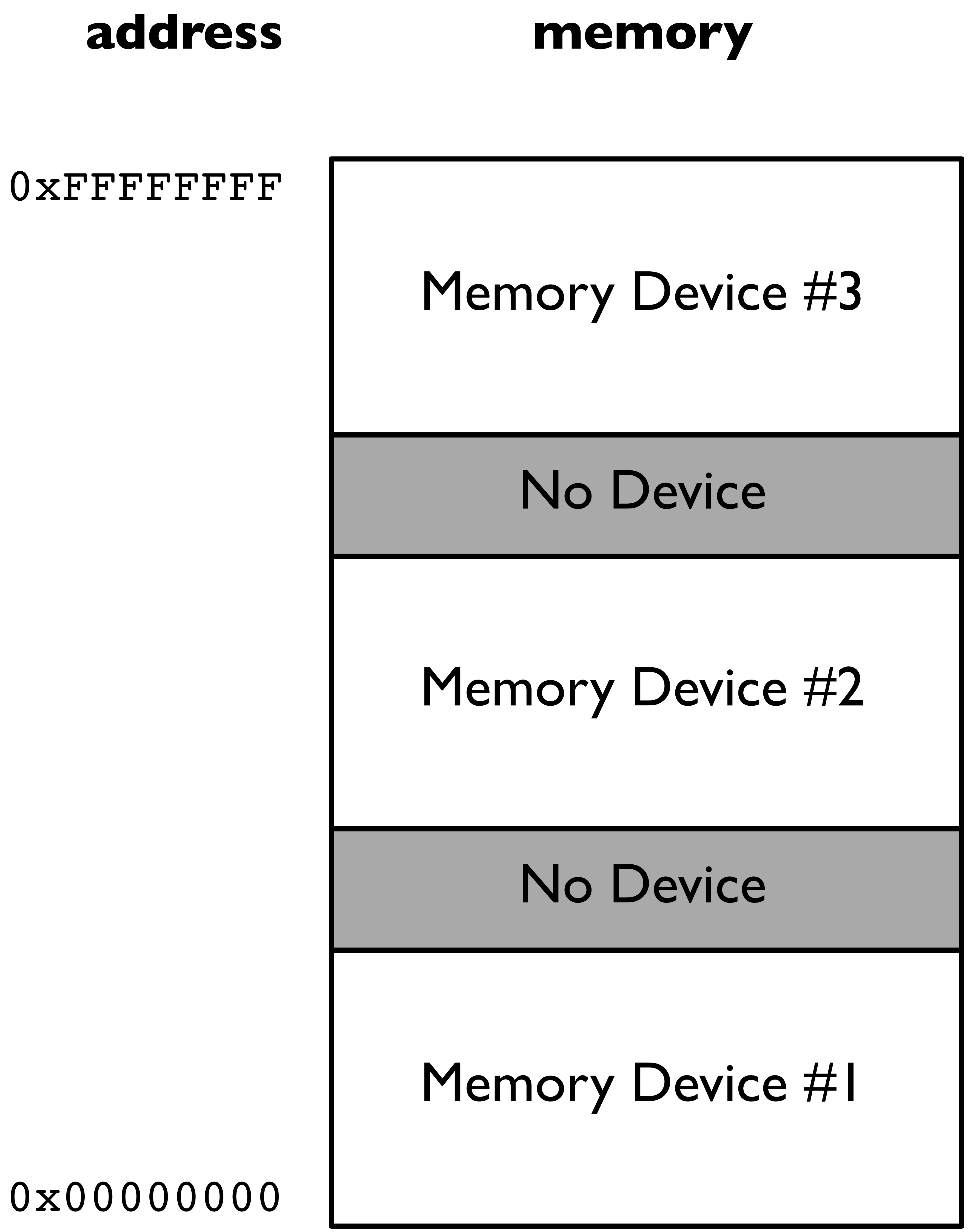




address bus
A[31:0]
control
address register
incrementer
PC
PC
register bank (R0 … R15)
multiplier
A bus
ALU bus
B bus
barrel
shifter
ALU
instruction
decode
and
control
logic
data out register
data in register
data bus
D[31:0]

Address Space and Memory Map





NXP LPC2468

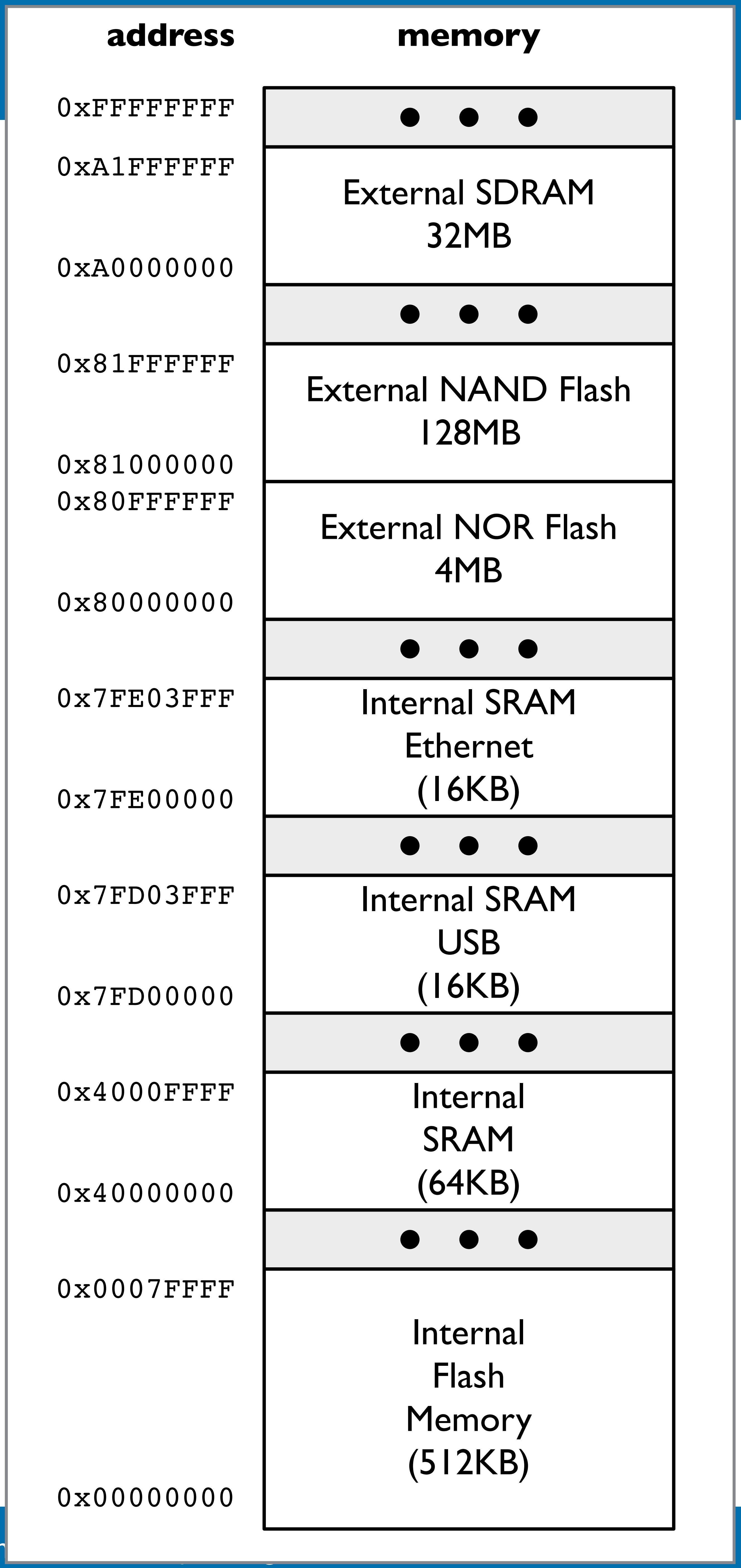
LPC2468 Development Board

- On-Chip Flash (Read-Only) Memory (512KB)
- On-Chip RAM
  (64KB + 16KB for Ethernet + 16KB = 96KB)
- Off-Chip RAM (32MB)

4GB address space (32 bit addresses)

Each memory device is mapped into a region of the address space

Memory accesses (loads/stores) are directed to the device that is mapped into the address being accessed



Memory Mapping Example

512KB Internal Flash Memory

512KB = 524288 bytes = 2^{19} bytes

Address range: 0 … 524287_{10} = 0x00000 … 0x7FFFF

Device required 19-bit addresses

Choose address 0x00000000 as device base address

Mapped into processor address space 0x00000000 … 0x0007FFFF

Similarly for Internal SRAM, External SDRAM, …

Memory Mapping Example

64KB Internal SRAM

64KB = 65536 bytes = 2^{16} bytes

Address range: 0 … 65535_{10} = 0x0000 … 0xFFFF

Device requires 16-bit addresses

Choose address 0x40000000 as device base address

Mapped into processor address space 0x40000000 … 0x4000FFFF

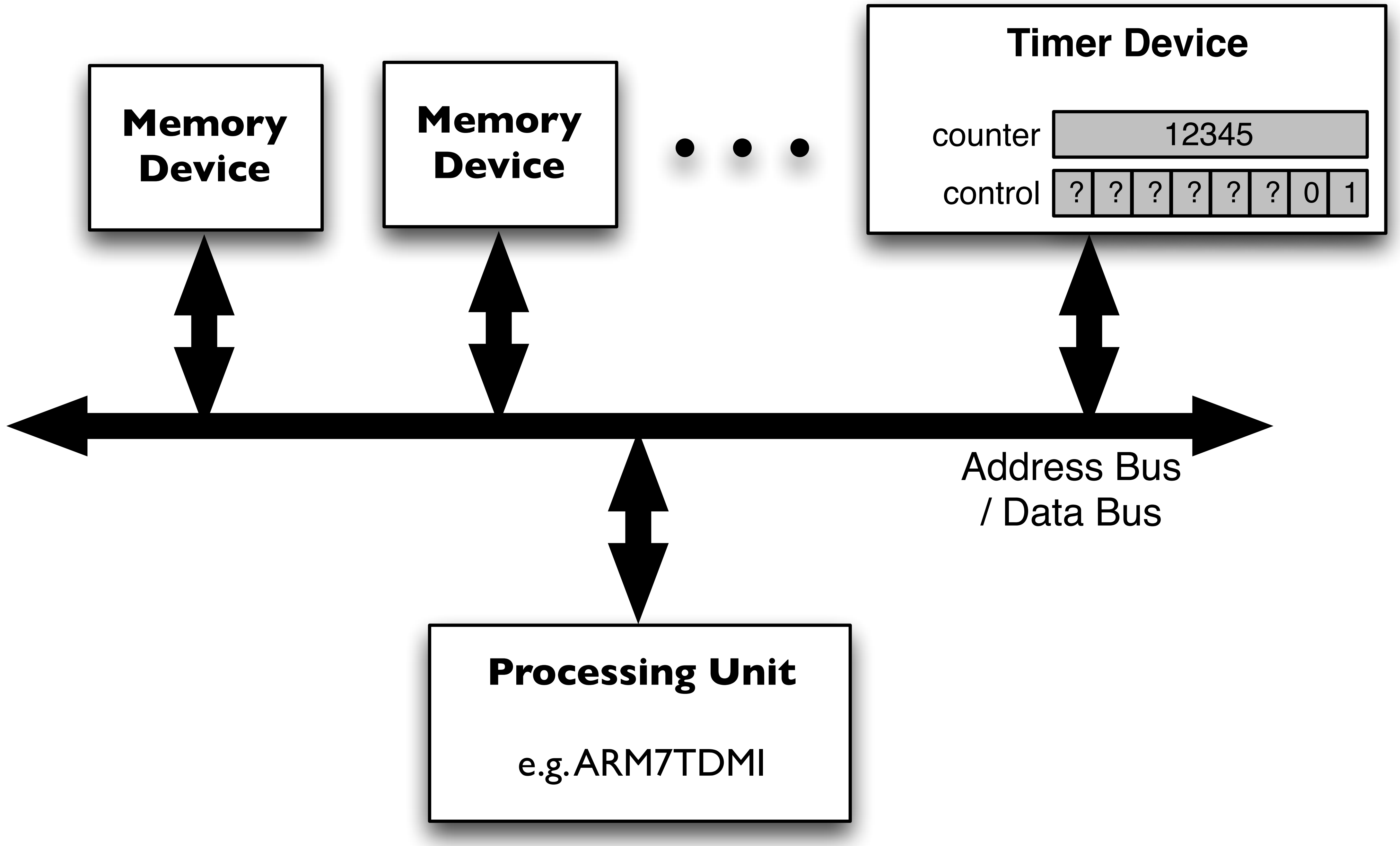
Other non-memory devices?

How do we communicate with non-memory ***devices*** external to the microprocessor?

Consider a ***fictional*** timer device that can measure elapsed time

- Fictional timer has two internal registers
- Control Register, 32-bits
  - bit 0: start/stop
  - bit 1: reset
  - bits 2-31: unused
- Counter Register, 32-bits, stores elapsed time in ms

Map device registers into the memory space (just like other "regular" memory devices!!)



Other non-memory Devices?

Choose a base address for our fictional timer peripheral

e.g. 0xE0004004

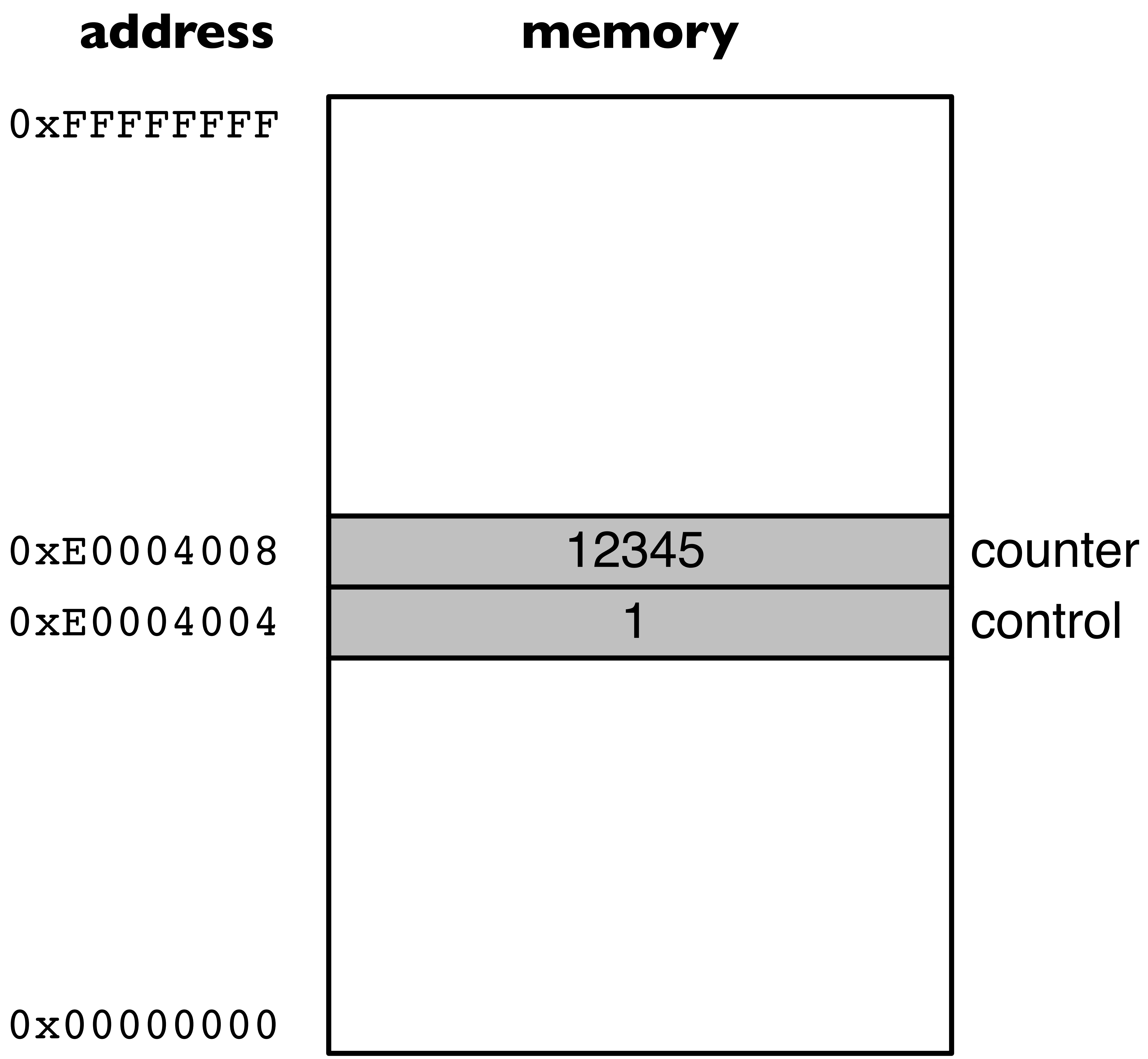
Map device registers to memory locations beginning at base address

control register: 0xE0004004

counter register: 0xE0004008

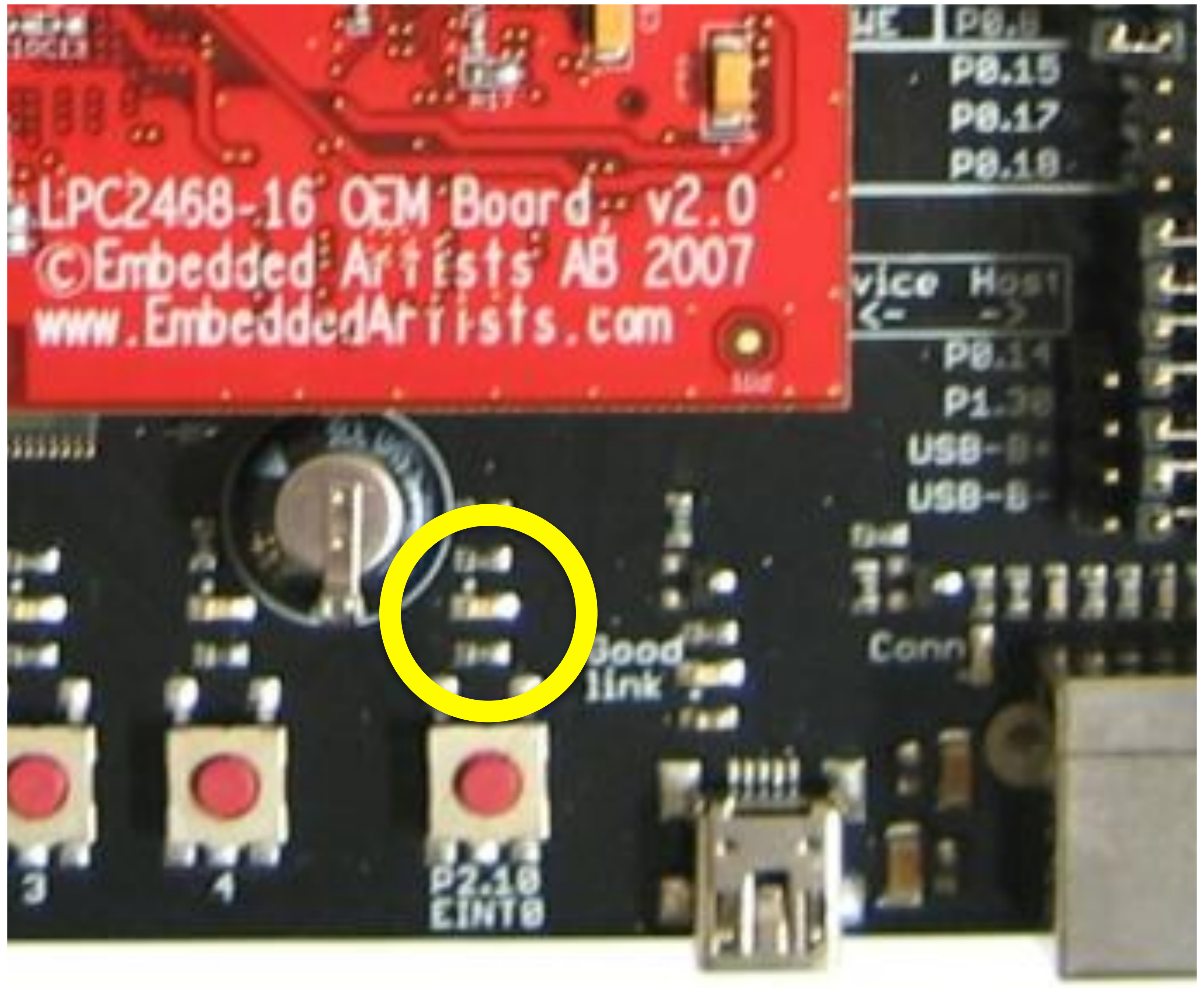
e.g. to reset timer

```
LDR   R4, =0xE0004004
LDR   R5, =2
STR   R5, [R4]
```

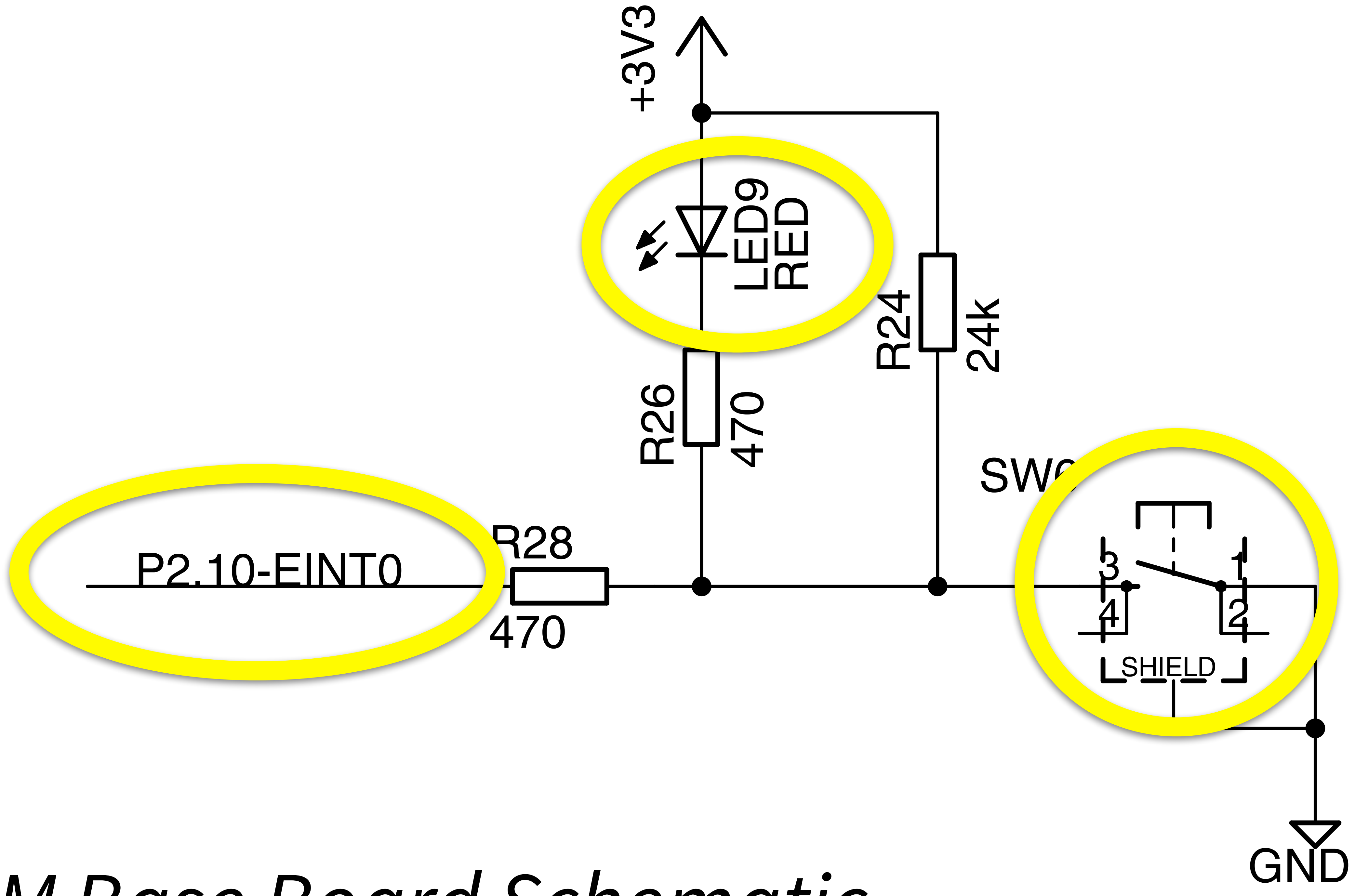


I/O Example

Design and write an ARM Assembly Language program that will cause an LED to blink on and off repeatedly



Interrupt (P2.10) key



LPC2468 OEM Base Board Schematic

Many external LPC2468 pins have multiple uses

Functionality of a pin is configured …

by software, at runtime

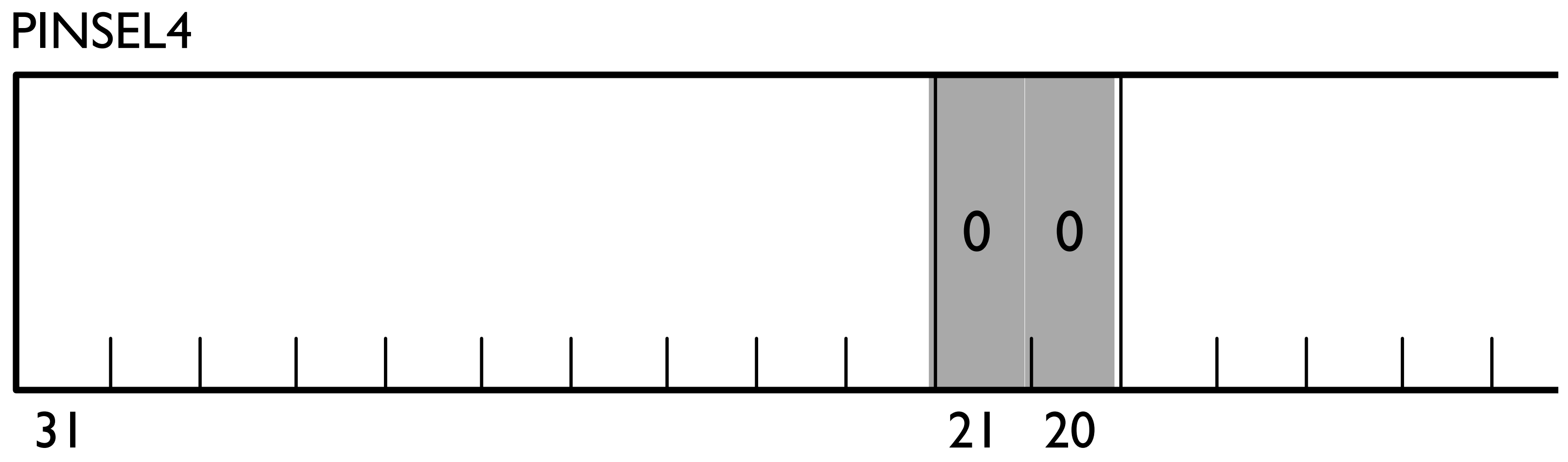
using the *Pin Connect Block* **PINSELx** memory mapped register

PINSELx defines pin function

Each 32-bit register controls 16 pins
(2 bits to select one of $2^2 = 4$ possible functions for each physical pin)

Table 135. LPC2420/60/68/70/78 pin function select register 4 (PINSEL4 - address 0xE002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
21:20	P2[10]	GPIO Port 2.10	$\overline{\text{EINT0}}$	Reserved	Reserved	00



LPC2468 User Manual
Chapter 9: LPC24xx Pin Connect

Table 135. LPC2420/60/68/70/78 pin function select register 4 (PINSEL4 - address 0xE002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2[0]	GPIO Port 2.0	PWM1[1]	TXD1	TRACECLK[1]/ LCDPWR	00
3:2	P2[1]	GPIO Port 2.1	PWM1[2]	RXD1	PIPESTAT0[1]/ LCDLE	00
5:4	P2[2]	GPIO Port 2.2	PWM1[3]	CTS1	PIPESTAT1[1]/ LCDDCLK	00
7:6	P2[3]	GPIO Port 2.3	PWM1[4]	DCD1	PIPESTAT2[1]/ LCDFP	00
9:8	P2[4]	GPIO Port 2.4	PWM1[5]	DSR1	TRACESYNC[1]/ LCDENAB/ LCDM	00
11:10	P2[5]	GPIO Port 2.5	PWM1[6]	DTR1	TRACEPKT0[1]/ LCDLP	00
13:12	P2[6]	GPIO Port 2.6	PCAP1[0]	RI1	TRACEPKT1[1]/ LCDVD[0]/ LCDVD[4]	00
15:14	P2[7]	GPIO Port 2.7	RD2	RTS1	TRACEPKT2[1]/ LCDVD[1]/ LCDVD[5]	00
17:16	P2[8]	GPIO Port 2.8	TD2	TXD2	TRACEPKT3[1]/ LCDVD[2]/ LCDVD[6]	00
19:18	P2[9]	GPIO Port 2.9	$\overline{\text{USB\_CONNECT1}}$	RXD2	EXTIN0[1]/ LCDVD[3]/ LCDVD[7]	00
21:20	P2[10]	GPIO Port 2.10	$\overline{\text{EINT0}}$	Reserved	Reserved	00
23:22	P2[11]	GPIO Port 2.11	$\overline{\text{EINT1}}$/ LCDCLKIN	MCIDAT1	I2STX_CLK	00
25:24	P2[12]	GPIO Port 2.12	$\overline{\text{EINT2}}$/ LCDVD[4]/ LCDVD[3]/ LCDVD[8]/ LCDVD[18]	MCIDAT2	I2STX_WS	00
27:26	P2[13]	GPIO Port 2.13	$\overline{\text{EINT3}}$/ LCDVD[5]/ LCDVD[9]/ LCDVD[19]	MCIDAT3	I2STX_SDA	00
29:28	P2[14]	GPIO Port 2.14	$\overline{\text{CS2}}$	CAP2[0]	SDA1	00
31:30	P2[15]	GPIO Port 2.15	$\overline{\text{CS3}}$	CAP2[1]	SCL1	00

Enable GPIO mode for Pin P2.10

Pin P2.10 possible functions

- GPIO port P2.10
- External interrupt source EINT0

Configured using bits 21:20 of PINSEL4

- Set bits 21:20 to 00_2 for GPIO function
- **REMEMBER!** Need to leave the other bits (0 … 19 and 22 … 31) of PINSEL4 unmodified
- PINSEL4 address – 0xE002C010 (from LPC2468 User Manual)

Read-Modify-Write operation to set register value

```
    ; Enable P2.10 for GPIO
    LDR   R5, =PINSEL4      ; 0xE002C010
    LDR   R6, [R5]          ; Read current PINSEL4 value
    BIC   R6, #(0x3 << 20)  ; Modify to clear bits 21:20
    STR   R6, [R5]          ; Write new PINSEL4 value
```

Set Pin P2.10 for output

GPIO pins can be either inputs or outputs

Controlling LED = output

Direction (I/O) set using **FIOxDIRy** register

- Use FIO2DIR1 for P2.10
- Set bit 2 of FIO2DIR1 to 1 for output
- Refer to LPC2468 User Manual

```
; Set P2.10 for output
LDR   R5, =FIO2DIR1
LDRB  R6, [R5]          ; Read FIO2DIR1
ORR   R6, #(0x1 << 2)   ; Set bit 2 to value 1 to configure output
STRB  R6, [R5]          ; Write FIO2DIR1
```

Repeatedly invert state of LED

Set output value (0/1) using bit 2 of FIO2PIN1 register

- Must not change other bits of FIO2PIN1

Read, Modify, Write again

- test if LED is on or off [READ]
- if it is off then turn it on, if it is on then turn it off [MODIFY]
- output new value [WRITE]

Naïve delay implemented using a counting loop

```
repeat                          ; while (forever) {
   LDR    R4, =0x04             ;   setup bit mask for bit 2 of FIO2 (P2.10)
   LDR    R5, =FIO2PIN1
   LDRB   R6, [R5]              ;   read FIO2PIN1
   AND    R7, R6, R4            ;   only want to test bit 2 — mask other bits
   CMP    R7, #0                ;   if (LED on)
   BNE    elseoff               ;   {
   BIC    R6, R6, R4            ;     clear bit 2 (turn LED off)
   B      endif                 ;   }
elseoff                         ;   else if 1, turn LED off {
   ORR    R6, R6, R4            ;     set bit 2 (turn LED on)
endif                           ;   }
   STRB   R4, [R5]              ;   write FIO2PIN1

   ; delay
   LDR    R4, =0x400000         ;   count = 0x400000
whDelay                         ;   while (count > 0)
   CMP    R4, #0                ;   {
   BEQ    eWhDelay              ;
   SUB    R4, R4, #1            ;     count = count - 1
   B      whDelay               ;   }
eWhDelay
   B      repeat                ; }
```