Basic Concepts

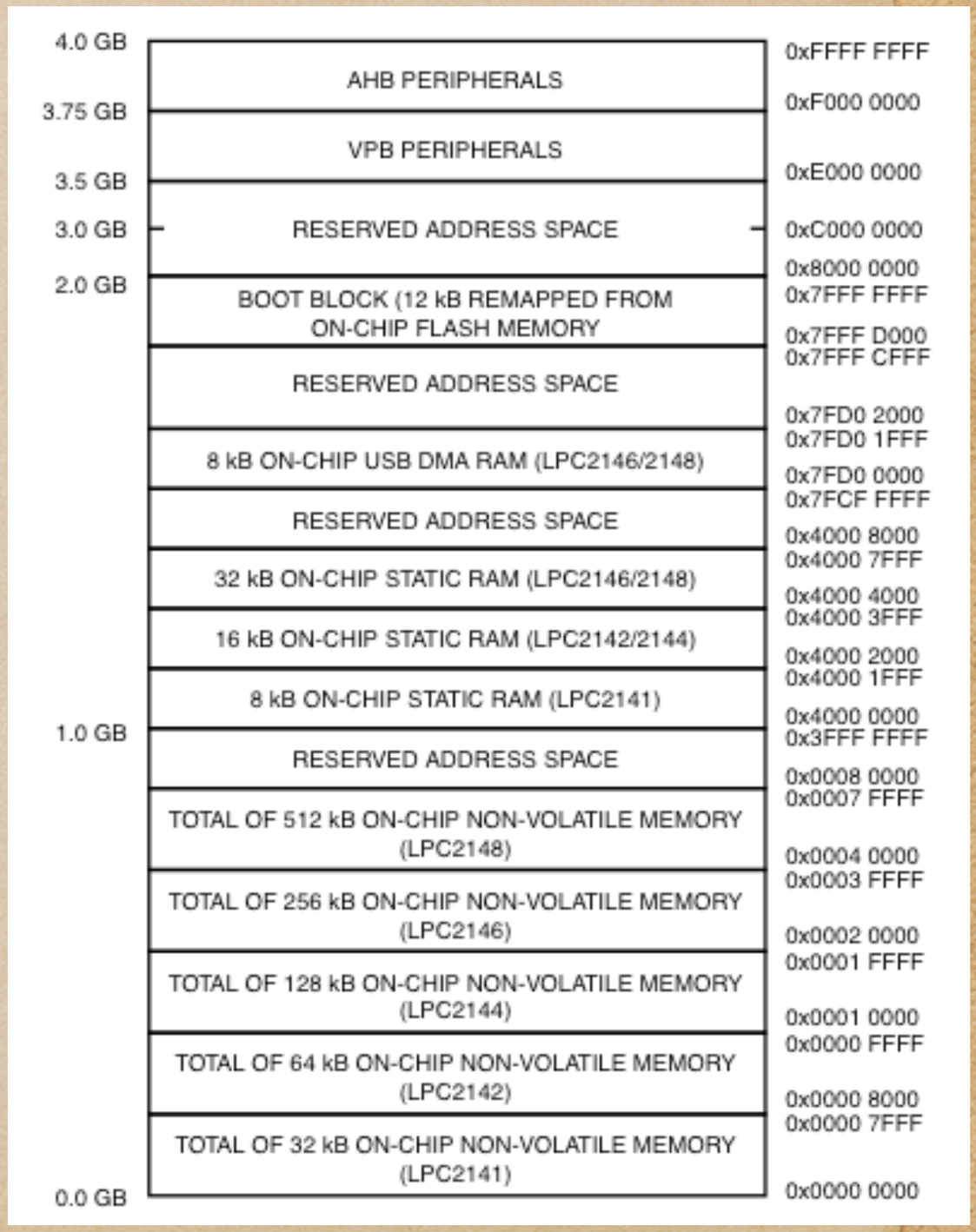
CECS 347

CSULB

Fall 2012

Memory Map

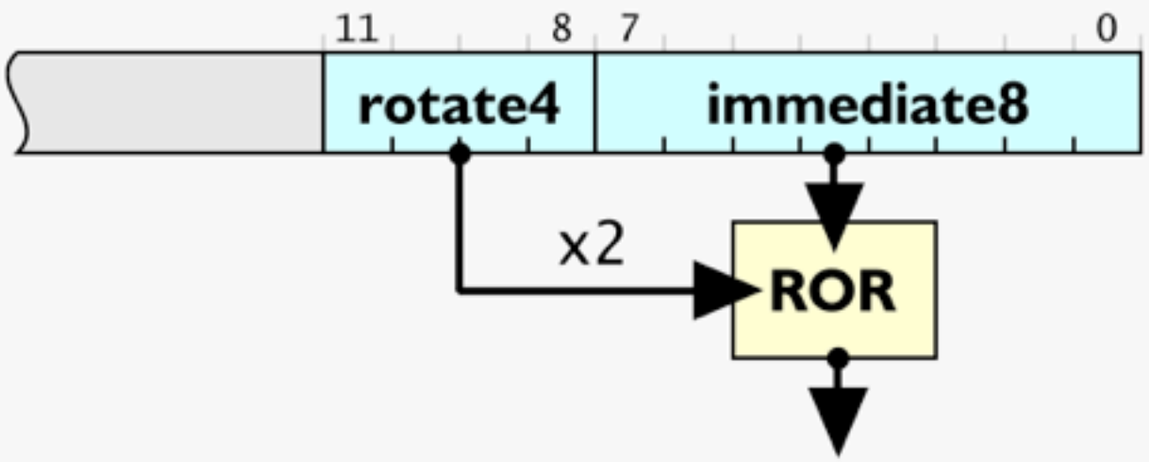
- In computer science, a memory map is a structure of data that indicates how the address space of the processor is partitioned
- This is applicable to any processor design but it is of particular importance in embedded systems design
- A system on a chip (SOC) will include flash memory, SRAM, configuration registers and I/O. Each of these "spaces" require associated addresses for the processor to address
- In particular in the LPC2148 the memory space is 4G bytes and is allocated according to the attached table



ARM7 Instructions

- All ARM7 instructions must fit in32 bit boundaries (four bytes)
- This means that the hardware must be able to obtain all information required to execute an instruction from the fields contained therein
- Consideration of the attached ARM instruction formats will reveal that the maximum immediate data may be is *8* bits

You can't fit an arbitrary 32-bit value into a 32-bit instruction word. ARM data processing instructions have 12 bits of space for values in their instruction word. This is arranged as a four-bit rotate value and an eight-bit immediate value:



The 4-bit rotate value stored in bits 11-8 is multiplied by two giving a range of 0-30 in steps of two.

Using this scheme we can express immediate constants such as:

- 0x000000FF
- 0x00000FF0
- 0xFF000000
- 0xF000000F

- The assembler is able to recognize the pseudoinstruction `LDR R0, =0x12345678` and will then utilize data storage and a sequence of ARM instructions to accomplish the load of the register with a 32 bit value

ARM7 Immediate Data Work Arounds

Loading Wide Values

You can form constants wider than those available in a single instruction by using a sequence of instructions to build up the constant. For example:

```
MOV r2, #0x55              ; R2 = 0x00000055
ORR r2, r2, r2, LSL #8     ; R2 = 0x00005555
ORR r2, r2, r2, LSL #16    ; R2 = 0x55555555
```

...or load the value from memory:

```
LDR r2, =0x55555555
```

The pseudo-instruction `LDR Rx,=const` tries to form the constant in a single instruction, if possible, otherwise it will generate an `LDR`.

- The simplest solution is to use the pseudo-instruction `LDR Rx, =const`
- You may generate your own data values as demonstrated in the code sequence above
- Thanks to by David Thomas, © 2006-2012

Align Directive Explanation

ERROR!

Address	Byte 3	Byte 2	Byte 1	Byte 0
00000000	Instruction 1			
00000004	Instruction 2			
00000008	Data	Data	Data	Data
0000000C	Data	Data	Instruction 3	
00000010	Instruction 3		Instruction 4	
00000014	Instruction 4		Data	Data

```
ADD R0, R1, R2
SUB R1, R3, R5
DCB 0x01,0x02,0x03,0x04,0x05,0x06
MUL R5, R2, R4
MOV R5, R5, LSL #1
DCB 0x07,0x08
```

NO ERROR

Address	Byte 3	Byte 2	Byte 1	Byte 0
00000000	Instruction 1			
00000004	Instruction 2			
00000008	Data	Data	Data	Data
0000000C	Data	Data	Fill	Fill
00000010	Instruction 3			
00000014	Instruction 4			

```
ADD R0, R1, R2
SUB R1, R3, R5
DCB 0x01,0x02,0x03,0x04,0x05,0x06
ALIGN
MUL R5, R2, R4
MOV R5, R5, LSL #1
DCB 0x07,0x08
```

- ALIGN will put the code to be programmed into the FLASH back on the proper boundary so the every instruction will be found on a word boundary and will thus avoid a prefetch abort

Assemblers

- An assembly language is a low-level programming language for a computer or microcontroller
- Each assembly language is particular to a particular computer architecture (ISA)
- An assembler is the tool that will translate the assembly language into executable machine code
- Assembly language uses a mnemonic to represent each low-level machine instruction (1:1 relationship)
- A multi-pass assembler will create a table with all symbols and their values on the first pass, and will then use the table to fill in the values on the second pass
- An assembler may include particular syntax to assist in the definition of the assembly language program
- A disassembled view of the generated object code (disassembly) will include all of the hex values contained in the machine code along with the assembly language to assist in the debug of the program

Compilers

- A compiler is a computer program that transforms source code written in a high-level programming language into object code (machine instructions)
- Some incarnations will compile the source code to assembly code to then be run through an assembler but this is the exception, not the rule
- Compilers provide the ability to generate *your* source code to be "target independent" or "portable" - it is the responsibility of the complier to generate the object code for a selected ISA
- The difference between a compiler and an interpreter is that the compiler will generate the object code to be loaded into the target computer and executed. An interpreter is a program that will parse the input source code and then perform the desired operations one by one
- The primary difference between compiled and interpreted code is speed
- There are also translators which will translate the source code into a form where the environment has the tools required to execute the code

Knowledge

- Knowledge is familiarity with someone or something, which can include facts, information, descriptions, or skills acquired through experience or education
- It can refer to theoretical or practical understanding of a subject
- It can be implicit (as with practical skill or expertise) or explicit (as with the theoretical understanding of a subject)
- Knowledge acquisition involves complex cognitive processes: perception, communication, association and reasoning
- Some pray "Favor us with knowledge, understanding and discretion"
- Fight to not let your engineering education just be an endless stream of tasks to perform. Fight to emerge with not only the accomplishment but also with the understanding of what you have accomplished