Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C

Chapter 5
Memory Access Exercises

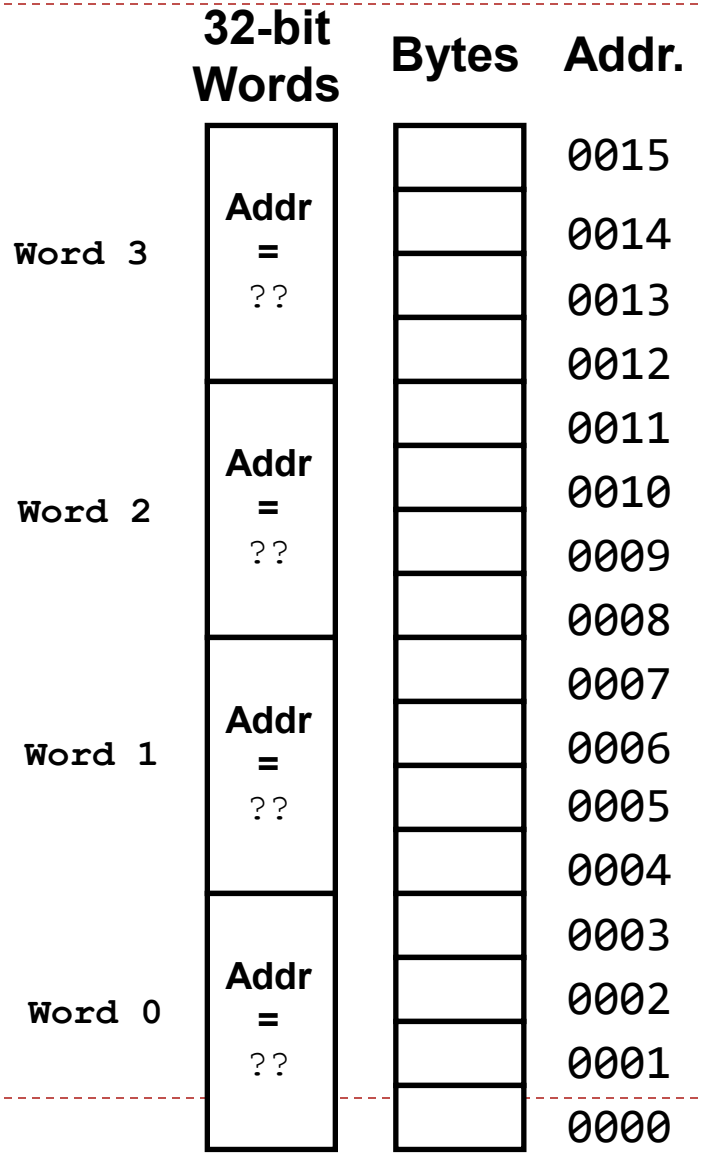
Zonghua Gu

Fall 2025

Acknowledgement: Lecture slides based on Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, University of Maine https://web.eece.maine.edu/~zhu/book/

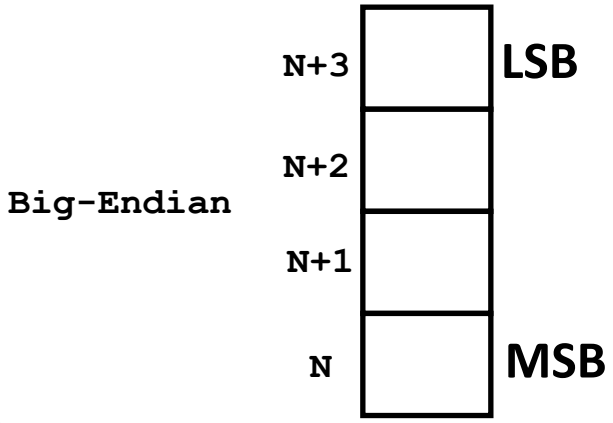
Endianness

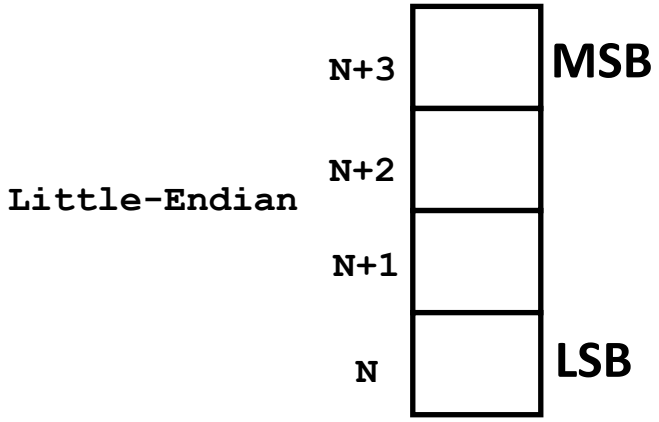
What are the memory address of these four words?



Endianness

- Q: Assume Big-Endian ordering. If a 32-bit word resides at memory address N, what is the address of:
  - (a) The MSB (Most Significant Byte)
  - (b) The 16-bit half-word corresponding to the most significant half of the word
- Q: Redo the question assuming Little-Endian ordering.





Endianness

The word stored at address 0x20008000 with Big-Endian ordering is

?

The word stored at address 0x20008000 with Little-Endian ordering is

?

Memory Address	Memory Data
0x20008003	0xA7
0x20008002	0x90
0x20008001	0x8C
0x20008000	0xEE

Endianness

```
LDR r11, [r0]
; r0 = 0x20008000
```

r11 before load

0x12345678

r11 after load w/ Big-Endian ordering

r11 after load w/ Little-Endian ordering

Memory Address	Memory Data
0x20008003	0xA7
0x20008002	0x90
0x20008001	0x8C
0x20008000	0xEE

Endianness

Memory Address	Memory Data
0x20008003	0xA7
0x20008002	0x90
0x20008001	0x8C
0x20008000	0xEE

- Assume little endian for the following questions: r0 = 0x20008000
- **LDRH r1, [r0]**
  - r1 after load:
- **LDSB r1, [r0]**
  - r1 after load:
- **STR r1, [r0], #4**
  - Assume r1 = 0x76543210
  - r0 after store:
  - Memory content after store:
- **STR r1, [r0, #4]**
  - Assume r1 = 0x76543210
  - r0 after store:
  - Memory content after store:
- **STR r1, [r0, #4]!**
  - Assume r1 = 0x76543210
  - r0 after store:
  - Memory content after store:

Data Alignment

- Q: Assume a byte-addressable memory with a data bus that is 32 bits (4 bytes) wide. Consider 16 bytes of memory (addresses 0 to 15) arranged as four 32-bit words (4 bytes each). How many memory cycles are required to read each of the following from memory?
  - (a) A 2-Byte operand read from decimal address 5
  - (b) A 2-Byte operand read from decimal address 15
  - (c) A 4-Byte operand read from decimal address 10
  - (d) A 4-Byte operand read from decimal address 20

Data Alignment

Address 111	Address 110	Address 109	Address 108
Address 107	Address 106	Address 105	Address 104
Address 103	Address 102	Address 101	Address 100
Address 99	Address 98	Address 97	Address 96

- Q: Assume a byte-addressable memory with a data bus that is 32 bits (4 bytes) wide. Consider 16 bytes of memory (addresses 0 to 15) arranged as four 32-bit words (4 bytes each).
  - (a) What is the address of MSB of the word at address 102, assuming Little-Endian ordering?
  - (b) What is the address of LSB of the word at address 102, assuming Little-Endian ordering?
  - (b) How many memory cycles are required to read the word at address 102?
  - (c) How many memory cycles are required to read the half word at address 102?

Memory Cycles

Address 15	Address 14	Address 13	Address 12
Address 11	Address 10	Address 9	Address 8
Address 7	Address 6	Address 5	Address 4
Address 3	Address 2	Address 1	Address 0

- Q: Assume a byte-addressable memory with a data bus that is 32 bits (4 bytes) wide.
  - It takes _____ memory cycle(s) to read a Byte from memory
  - It takes _____ memory cycle(s) to read a half-word from memory
  - It takes _____ memory cycle(s) to read a word from memory
  - It takes _____ memory cycle(s) to read a double word from memory

Arrays

- Q: If the first element of a one-dimensional array x[] is stored at memory address 0x12345678, what is address of the second element if the array x[] contains
  - (a) chars
  - (b) shorts
  - (c) ints
  - (d) longs

LDM

- Assume that memory and registers r0 through r3 appear as follows. Suppose r3 = 0x8000. Describe the memory and register contents after executing each instruction (individually, not sequentially):
  - LDMIA r3!, {r0, r1, r2}
  - Or LDMIB r3!, {r2, r1, r0}
  - Or LDMIB r3!, {r1, r2, r0}

	Memory Address	Memory Data
	0x8010	0x00000001
	0x800c	0xFEEDDEAF
	0x8008	0x00008888
	0x8004	0x12340000
r3 ⇨	0x8000	0xBABE0000

LDR

- Suppose R2 and R5 hold the values 8 and 0x23456789 After following code runs with big-endian ordering, what value is in R7? How about with little-endian ordering?
  - STR R5, [R2, #0]
  - LDRB R7, [R2, #1]
  - LDRSH R7, [R2, #1]
  - LDRSH R7, [R2, #2]

Program Understanding 1

- Compute register and memory values at each step of this program, given initial register values and memory contents, assuming little-endian ordering. (Memory addresses increase from top to bottom, and from left to right in the table.)

```
MOVW R0, #0xAFE1
MOVT R0, #0xBADC
MOVT R2, #0xABCD
STR R3, [R1]
LDRSH R4, [R1, #0xC]
```

Initial Register Values

R0	0x00000000
R1	0x10000200
R2	0x0000FFFF
R3	0x18675309
R4	0x00000000
R5	0x00000000
…	
R13	0x10000200

0x10000200	60	1B	11	12	EE	FF	11	22	33	44	55	66	77	88	99	92
0x100001F0	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
0x100001E0	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F

Initial Memory Contents

Program Understanding 2

- Show all updates to registers as the assembly code shown below runs, assuming little-endian ordering. (Memory addresses increase from top to bottom, and from left to right in the table.) Show the NZCV flags next to the instruction if they change. Each instruction is 32 bits.

```
     LDR R1, =0x10000010
     LDR R2, [R1]
     LDR R3, [R1,#4]
     BL max
     SUBS R4, R2, R0  @NZCV =
     MOVW R5, #1
     LSL R6, R5, #4
     BIC R7, R2, R6
     ANDS R8, R7, R6  @NZCV =
     ROR R9, R3, #12
     REV R10, R9
     RBIT R11, R10
     ADDS R12, R3, R9 @NZCV =
     STR R12, [R1,#8]

loop B loop
     ENDP

max PROC
     CMP R2, R3       @NZCV =
     BLT second
first MOV R0, R2
      B done
second MOV R0, R3
done   BX LR
     ENDP
```

R0		R4		R8		R12	
R1		R5		R9		R13	
R2		R6		R10		R14	
R3		R7		R11		R15	

0x10000010	FF	EF	CD	AB	00	00	CD	AB	00	00	00	00