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

Chapter 5
Memory Access Exercises ANS

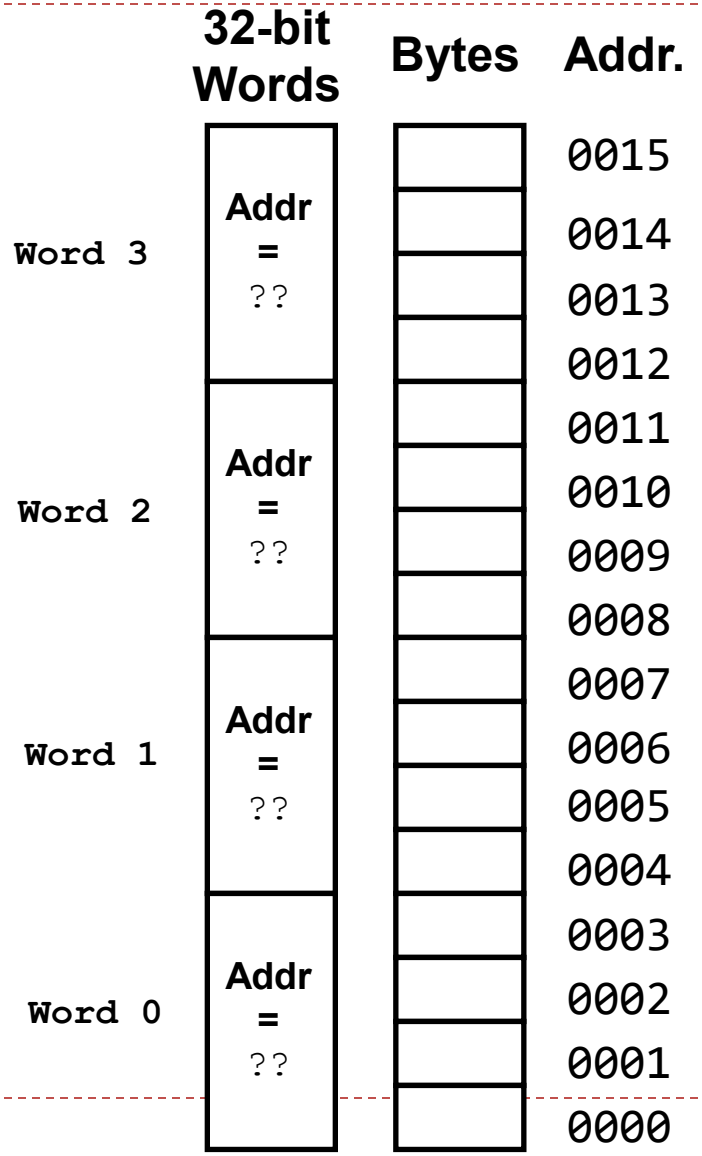
Zonghua Gu

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/

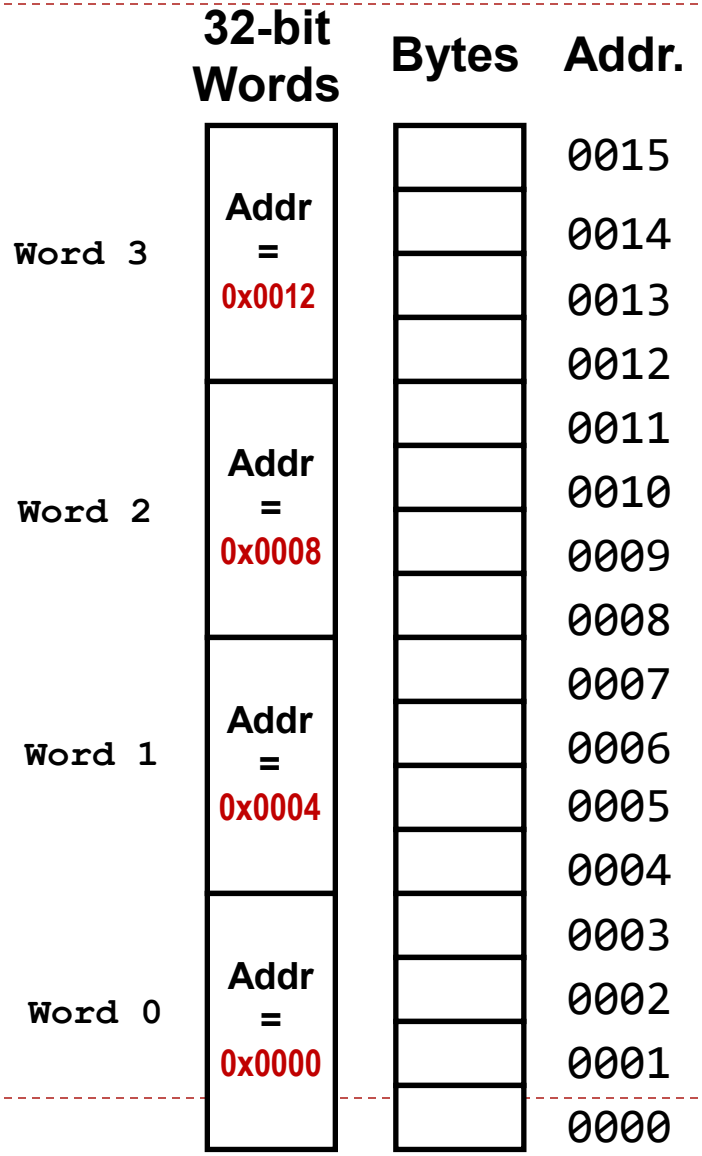
Question: Endianness

What are the memory address of these four words?



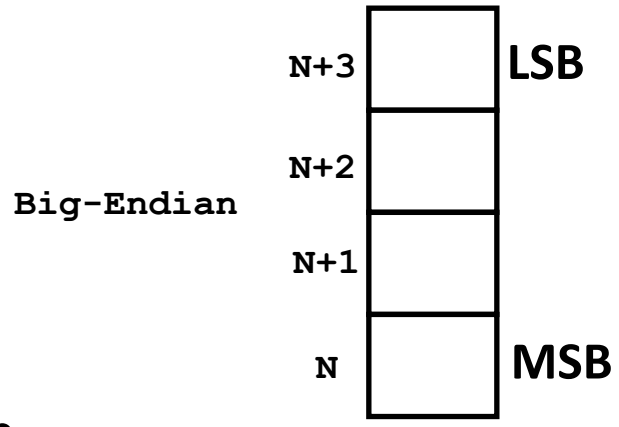
Answer: Endianness

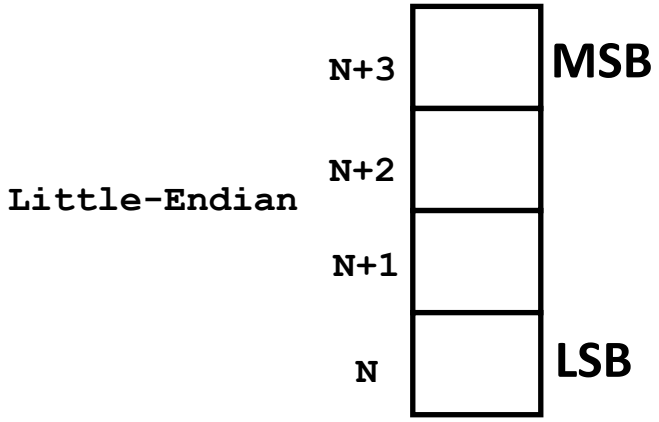
What are the memory address of these four words?
Same as the address of the lowest-address Byte
(this is true **for either Little-Endian or Big-Endian ordering**)



Question: Endianness

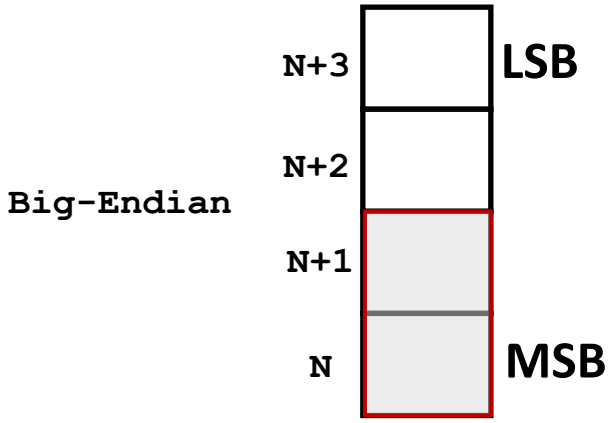
- Q: Assume Big-Endian ordering. If a 32-bit word resides at memory address N, what is the address of:
  - (a) The MSB (Most Significant Byte)
  - (b) The 16-bit half-word corresponding to the most significant half of the word
- Q: Redo the question assuming Little-Endian ordering.

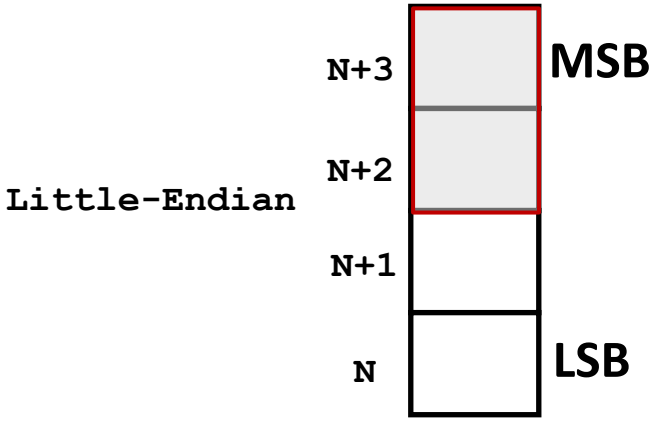




Answer: Endianness

- A: With Big-Endian ordering:
  - (a) Address of MSB: N
  - (b) Address of 16-bit half-word corresponding to the most significant half of the word: N (the half-word has address range of [N, N+1], so its address is N)
- With Little-Endian ordering:
  - (a) Address of MSB: N+3
  - (b) Address of 16-bit half-word corresponding to the most significant half of the word: N+2 (the half-word has address range of [N+2, N+3], so its address is N+2)





Question: 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

Answer: Endianness

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

0xEE8C90A7

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

0xA7908CEE

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

Endianness only specifies byte order, not bit order in a byte!

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 ANS

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

r11 before load

0x12345678

r11 after load w/ Big-Endian ordering

0xEE8C90A7

r11 after load w/ Little-Endian ordering

0xA7908CEE

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

Review

Data Alignment

- **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**

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

Well-aligned: each word begins on a mod-4 address, which can be read in a single memory cycle

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

Ill-aligned: a word begins on address 6, not a mod-4 address, which can be read in 2 memory cycles

The first read cycle would retrieve 4 bytes from addresses 4 through 7; of these, the bytes from addresses 4 and 5 are discarded, and those from addresses 6 and 7 are moved to the far right;
The second read cycle retrieves 4 bytes from addresses 8 through 11; the bytes from addresses 10 and 11 are discarded, and those from addresses 8 and 9 are moved to the far left;
Finally, the two halves are combined to form the desired 32-bit operand:

		Address 7	Address 6 (LSbyte)
Address 9 (MSbyte)	Address 8		
Address 9 (MSbyte)	Address 8	Address 7	Address 6 (LSbyte)

Question: 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

Answer: Data Align

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. 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
- A: (a) The operand contains memory content in address range [5,6]. It can be read in 1 memory cycle; the memory controller returns a word in address range [4,7]. The operand can be obtained via 1-Byte offset addressing into the word.
- (b) The operand contains memory content in address range [15,16]. It can be read in 2 memory cycles; the memory controller returns 2 words in address ranges [12,15] and [16, 19], which can be combined to return a word in address range [14,17]. The operand can be obtained via 1-Byte offset addressing into the word.
- (c) The operand contains memory content in address range [10,13]. It can be read in 2 memory cycles; the memory controller returns 2 words in address ranges [8,11] and [12, 15], which can be combined to return a word with address range [10,13].
- (d) The operand contains memory content in address range [20,23]. Since 20%4=0, it is well-aligned, and can be read in 1 memory cycle.

Question: Data Align

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?

Answer: Data Align

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?
- A:
  - (a) MSB of the word at address 102 is 105
  - (b) LSB of the word at address 102 is 102
  - (c) 2 cycles.
  - (d) 1cycle.

Answer: 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

Answer: 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
- A:
  - It takes __1__ memory cycle(s) to read a Byte from memory
  - It takes __1 or 2__ memory cycle(s) to read a half-word from memory
  - It takes __1 or 2__ memory cycle(s) to read a word from memory
  - It takes __2 or 3__ memory cycle(s) to read a double word from memory (a double word may span at most 3 consecutive words in memory)

Question: Arrays

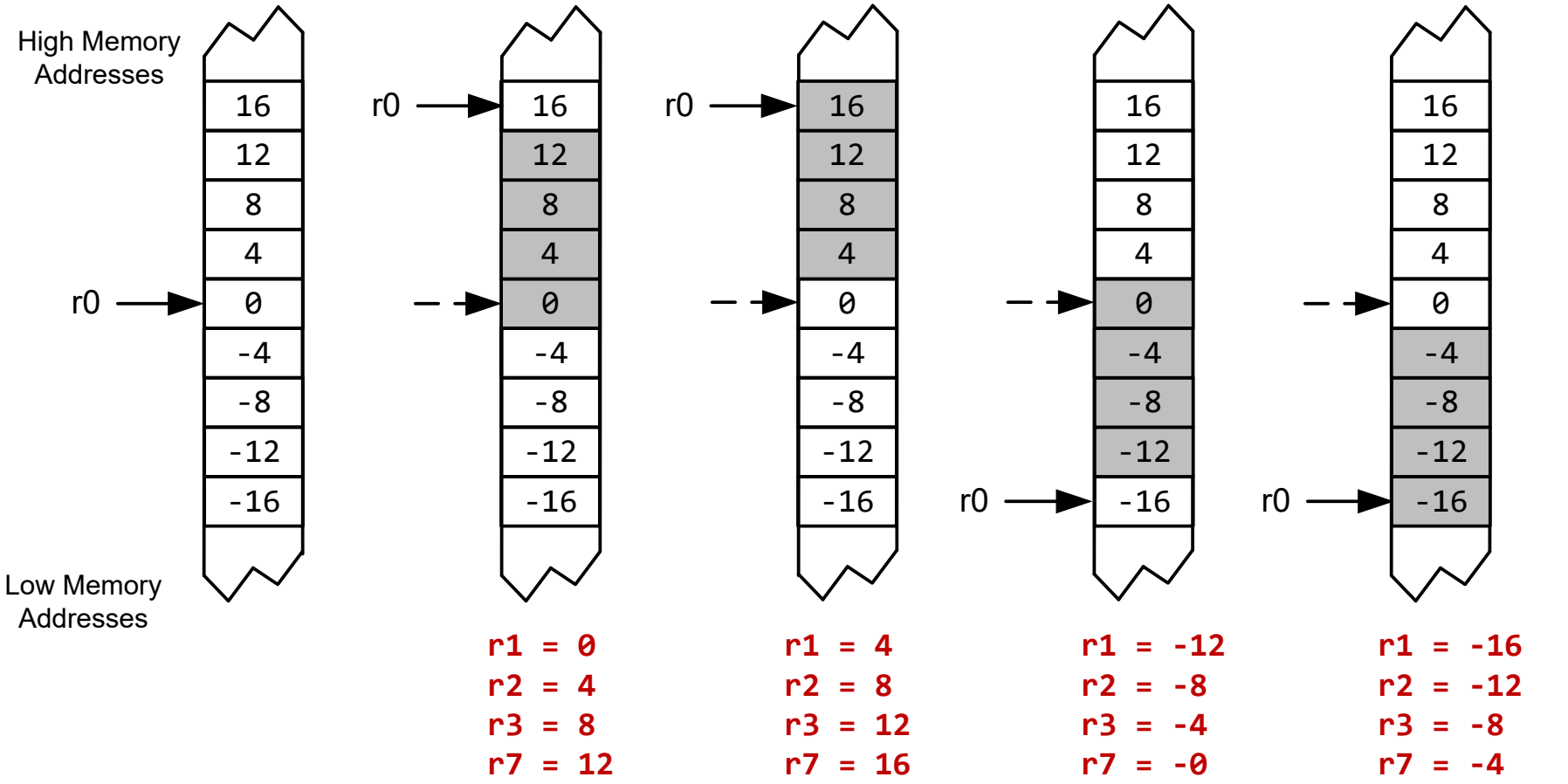
- Q: If the first element of a one-dimensional array x[] is stored at memory address 0x12345678, what is address of the second element if the array x[] contains
  - (a) chars
  - (b) shorts
  - (c) ints
  - (c) longs

Answer: 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
  - (c) longs
- A: x[1]'s address is x's address plus the data type size in Bytes
  - (a) chars: 0x12345678+1= 0x12345679
  - (b) shorts: 0x12345678+2= 0x1234567A
  - (c) ints: 0x12345678+4= 0x1234567C
  - (c) longs: 0x12345678+8= 0x12345680

Load Multiple Registers

LDMxx r0!, {r3,r1,r7,r2}



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}

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

LDM ANS

- 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}
- ANS:
  - After LDMIA r3!, {r0, r1, r2}
  - r0 = 0xBABE0000 (loaded from 0x8000)
  - r1 = 0x12340000 (loaded from 0x8004)
  - r2 = 0x00008888 (loaded from 0x8008)
  - r3 = 0x800C (auto-incremented)
  - Or after LDMIB r3!, {r2, r1, r0}
  - r0 = 0x12340000 (loaded from 0x8004)
  - r1 = 0x00008888 (loaded from 0x8008)
  - r2 = 0xFEEDDEAF (loaded from 0x800c)
  - r3 = 0x800C (auto-incremented)

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

- Assume that memory and registers r0 through r3 appear as follows. Describe the memory and register contents after executing the instruction:
  - LDMIA r3!, {r0, r1, r2}[1]