1. Computer HLE works right.

- `movl $(%ebp), %eax`
- `add $0x2007, %eax`

2. A `0000 2007` 4-byte, 32-bit word is stored in contiguous byte addresses in memory.

- `A`: The smallest byte address A
- `A+1`: (among A, A+1)
- `A+2`: (A+2, A+3)

3. The computer uses the byte address to access the whole word.

   - Big Endian
   - Little Endian
   - MSB - Most Significant Byte
   - LSB - Least Significant Byte

- Big Endian
- Little Endian

A addresses the little end.
4) C/C++ language rule:

Array elements at increasing indices are stored in memory at increasing addresses.

```
int IV = 0x2007;
unsigned char *p = (unsigned char*) & IV;
printf("%2x %2x %2x %2x\n", p[0], p[1], p[2], p[3]);
```

Prints

| 7 | 20 | 0 | 0 | 0 | 0 | 0 | 20 | 7 |

on a Little Endian computer

on a Big Endian computer

Here, we CASTED an int pointer to a char pointer. When *p is dereferenced (with *p or p[i]) a char will be accessed

```
(ia32 asm:  movl P, %edx  
            movb (%edx), %eax)
```
union {
    int Iway;
    unsigned char Away[4];
} MyUnion;

struct {
    int I;
    unsigned char A[4];
} MyStruct;

MyUnion, Iway        MyStruct, I
MyUnion, Away[0..3]  MyStruct, A[0..3]

3 names for ONE 4-byte chunk of memory
Structure with two distinct subvariables.

MyUnion, Away[0] refers to the byte in this chunk with the smallest address.

MyUnion, Iway = 0x2007;
if (MyUnion, Away[0] == 0x07)
    { printf("I'm Little Endian:\n"); }
else if (MyUnion, Away[0] == 0x00)
    { printf("I'm Big Endian:\n"); }
else
    { printf("I'm Broken: (\n"); }