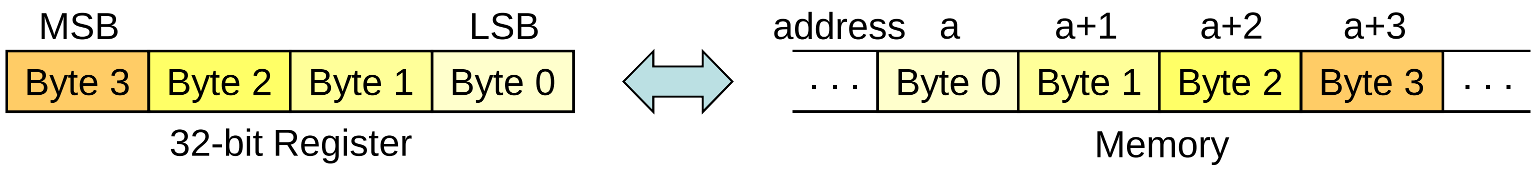
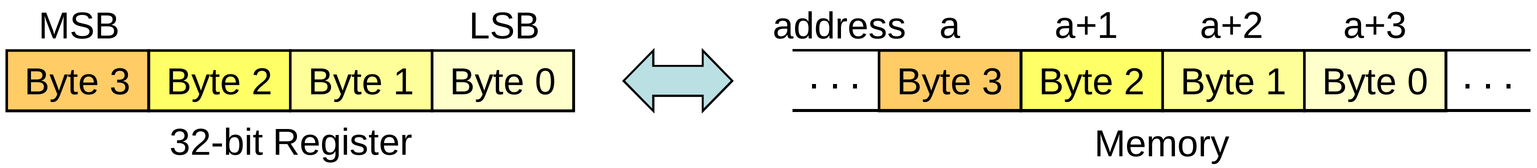
Byte Ordering and Endianness

- Processors can order bytes within a word in two ways
- Little Endian Byte Ordering
  - Memory address = Address of **least significant byte**
  - Examples: Intel 80x86



- Big Endian Byte Ordering
  - Memory address = Address of **most significant byte**
  - Examples: MIPS, Motorola 68k, SPARC



JMP Instruction

- JMP is an unconditional jump to a destination instruction
- Syntax: **`JMP destination`**
- JMP causes the modification of the EIP register

  EIP ← destination address
- A label is used to identify the destination address
- Example:

```
top:
    . . .
    jmp top
```

- JMP provides an easy way to create a loop
  - Loop will continue endlessly unless we find a way to terminate it

LOOP Instruction

- The LOOP instruction creates a counting loop
- Syntax: **`LOOP`** ***`destination`***
- Logic: ECX ← ECX – 1

  if ECX != 0, jump to *destination* label
- ECX register is used as a counter to count the iterations
- Example: calculate the sum of integers from 1 to 100

```
   mov  eax, 0      ; sum   = eax
   mov  ecx, 100    ; count = ecx
L1:
   add  eax, ecx    ; accumulate sum in eax
   loop L1          ; decrement ecx until 0
```

Your turn . . .

What will be the final value of EAX?

Solution: 10

```
    mov   eax,6
    mov   ecx,4
L1:
    inc   eax
    loop L1
```

How many times will the loop execute?

Solution: 2^{32} = 4,294,967,296

What will be the final value of EAX?

Solution: same value 1

```
    mov   eax,1
    mov   ecx,0
L2:
    dec   eax
    loop L2
```

Nested Loop

If you need to code a loop within a loop, you must save the outer loop counter's ECX value

```
.DATA
   count DWORD ?
.CODE
   mov ecx, 100     ; set outer loop count to 100
L1:
   mov count, ecx   ; save outer loop count
   mov ecx, 20      ; set inner loop count to 20
L2: .
    .
   loop L2          ; repeat the inner loop
   mov ecx, count   ; restore outer loop count
   loop L1          ; repeat the outer loop
```