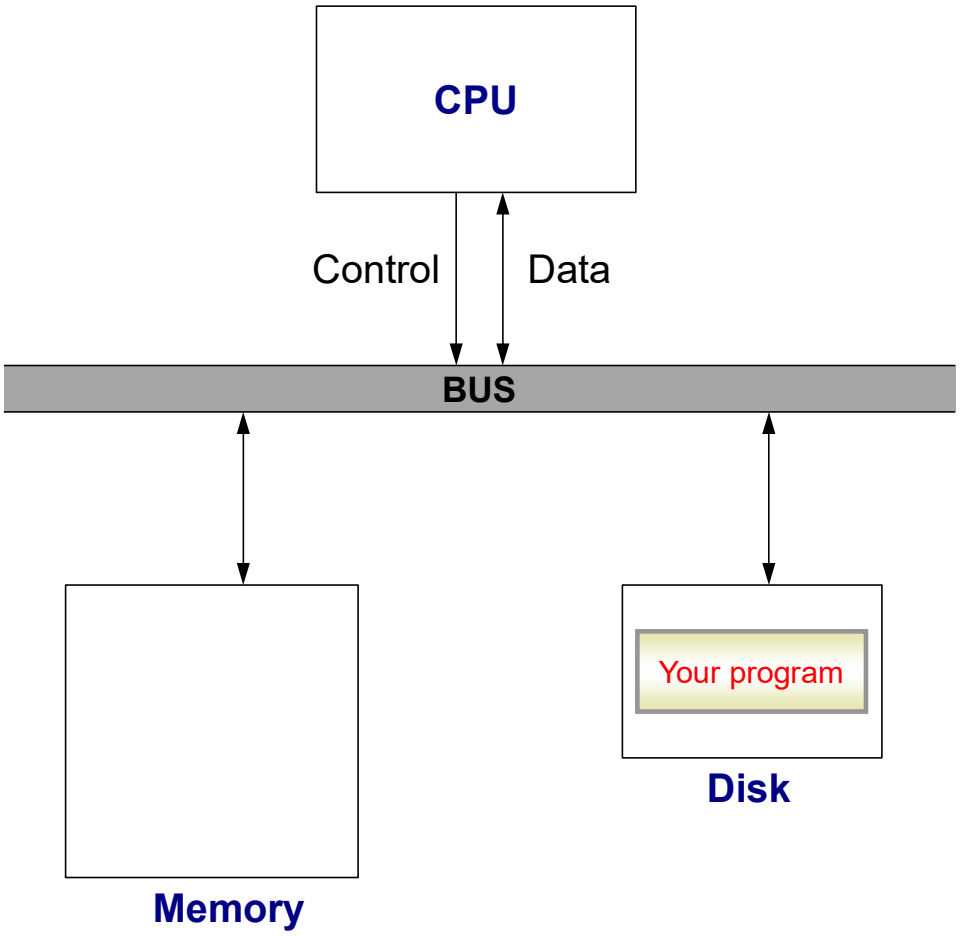
CSC 2400: Computer Systems

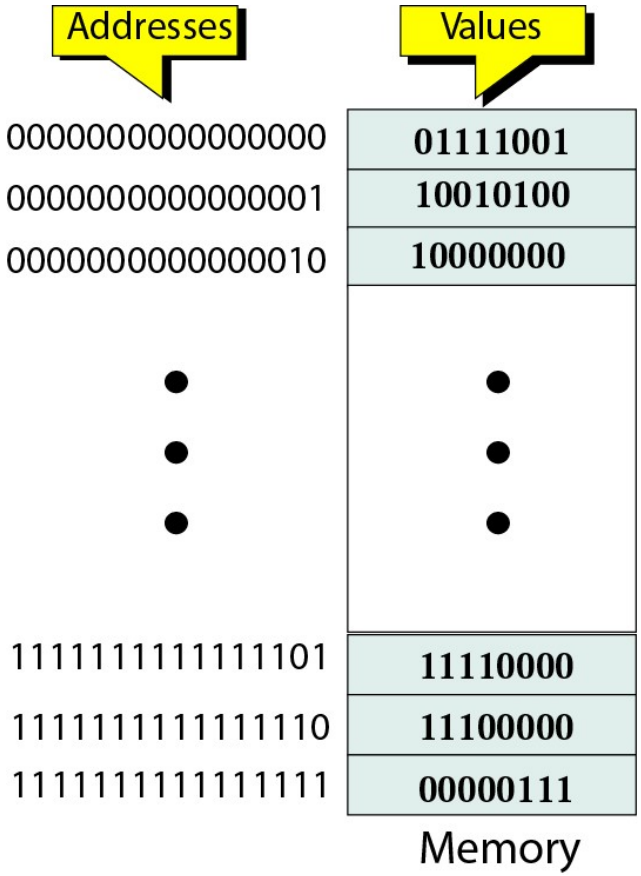
# Data Representation

# Computers and Programs

- A computer is basically a processor (CPU) interacting with memory
- Your program (executable) must be first loaded into memory before it can start executing



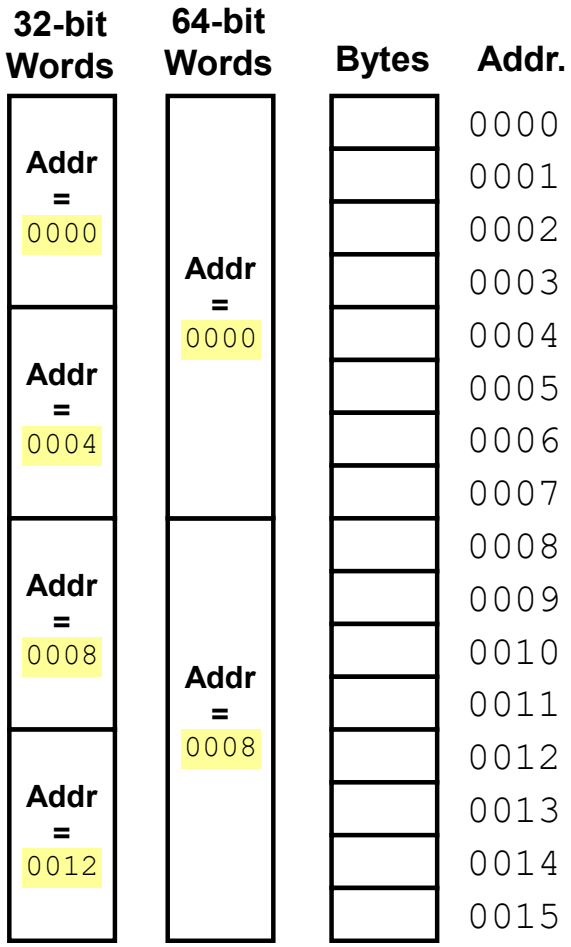
# Memory: Array of Bytes




- Memory is basically an array of bytes, each with its own address
- 1 byte = 8 bits
- Memory addresses are defined using ***unsigned binary integers***

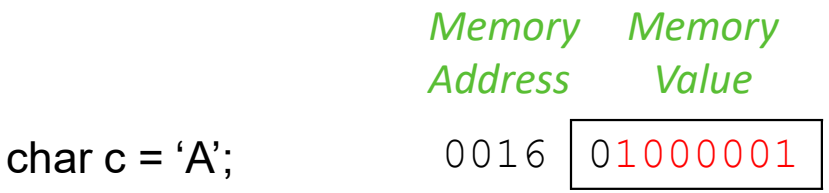
# Memory: Array of Words

- A *word* is a group of bytes handled as a unit by the CPU
  - tied to the CPU architecture
- Word address
  - address of first byte in word
  - addresses of successive words differ by 4 (32-bit) or 8 (64-bit)



# Memory and Variables

- What happens when you declare a variable?
  - The compiler allocates a memory box for that variable
  - How big a box?
    - Depends on the type of the variable



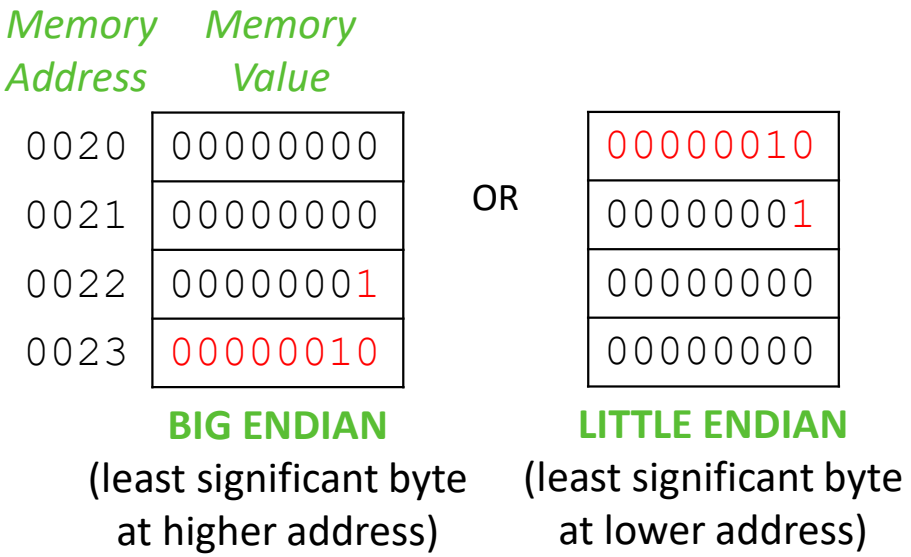
# One Annoying Thing: Byte Order

- Hosts differ in how they store data
  - E.g., four-byte number (byte3, byte2, byte1, byte0)
- Little endian (“little end comes first”) ← Intel PCs!!!
  - Low-order byte stored at the lowest memory location
  - Byte0, byte1, byte2, byte3
- Big endian (“big end comes first”)
  - High-order byte stored at lowest memory location
  - Byte3, byte2, byte1, byte 0
- Makes it more difficult to write portable code
  - Client may be big or little endian machine
  - Server may be big or little endian machine

# Memory and Variables (contd.)

int i = 258; | 00000000 00000000 00000001 00000010 |

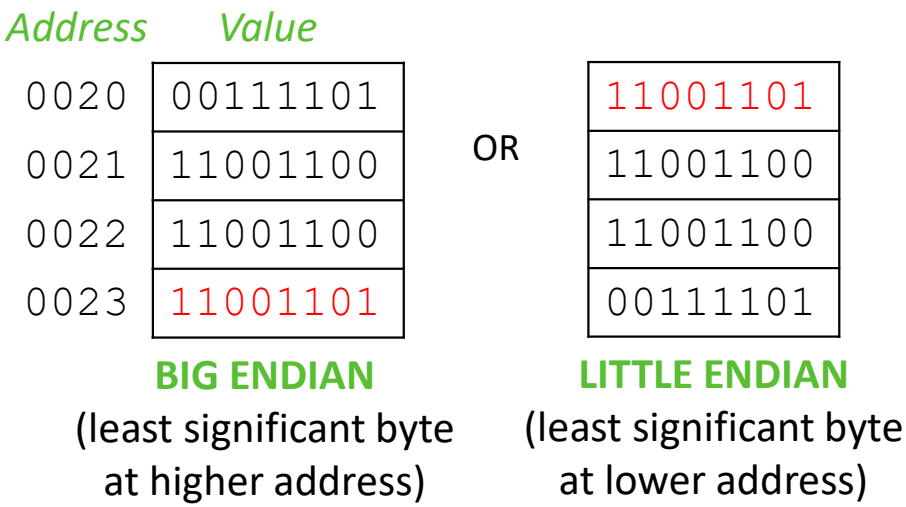
Memory view:



# Memory and Variables (contd.)

float f = 0.1; | 00111101 11001100 11001100 11001101 |

Memory view:



# Data Representations

- Sizes of C Data Types (in bytes)

| C Data Type | Sparc | Typical 32-bit | Intel IA32 |
|---|---|---|---|
| int | 4 | 4 | 4 |
| long int | 8 | 4 | 4 |
| char | 1 | 1 | 1 |
| short | 2 | 2 | 2 |
| float | 4 | 4 | 4 |
| double | 8 | 8 | 8 |
| long double | 8 | 8 | 10/12 |
| void * | 8 | 4 | 4 |

# The sizeof Operator

| Category | Operators |
|---|---|
| sizeof | `sizeof(type)`<br>`sizeof(expr)` |

- Unique among operators: evaluated at compile-time
- Evaluates to type size_t; on most systems, same as unsigned int
- Examples

```
int i = 10;
double d = 100.0;
…
… sizeof(int) …          /* On matrix, evaluates to 4 */
… sizeof(i) …            /* On matrix, evaluates to 4 */
… sizeof(double)…        /* On matrix, evaluates to 8 */
… sizeof(d) …            /* On matrix, evaluates to 8 */
… sizeof(d + 200.0) …    /* On matrix, evaluates to 8 */
```

# Determining Data Sizes

- Program to determine data sizes on your computer

```
#include <stdio.h>

int main()
{
   printf("char:        %d\n", (int)sizeof(char));
   printf("short:       %d\n", (int)sizeof(short));
   printf("int:         %d\n", (int)sizeof(int));
   printf("long:        %d\n", (int)sizeof(long));
   printf("float:       %d\n",  ____________________);
   printf("double:      %d\n",  ____________________);
   printf("long double: %d\n",  ____________________);
   return 0;
}
```

- Output on matrix

```
char:        1
short:       2
int:         4
long:        4
float:       4
double:      8
long double: 16
```