X

Physical Memory And Physical Addressing

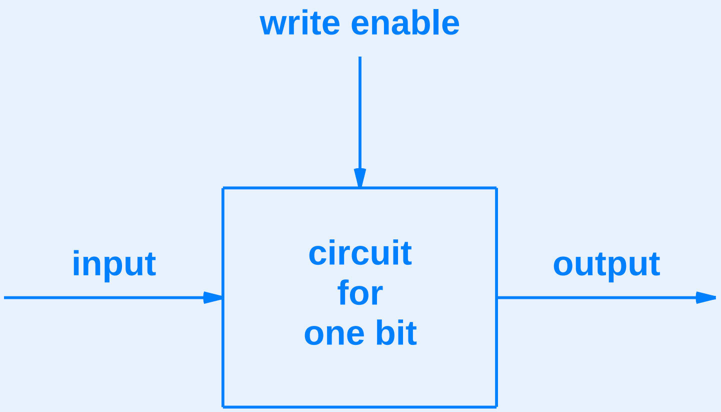
Computer Memory

- Main memory known as *Random Access Memory* (*RAM*)
- Usually volatile
- Two basic technologies available
  - Static RAM
  - Dynamic RAM

Static RAM (SRAM)

- Easiest to understand
- Similar to flip-flop

Illustration Of Static RAM



- When *enable* is high, output is same as input
- Otherwise, output holds last value

Advantages And Disadvantages Of SRAM

- Chief advantage
  - High speed
  - No extra refresh circuitry required
- Chief disadvantages
  - Power consumption
  - Heat
  - High cost

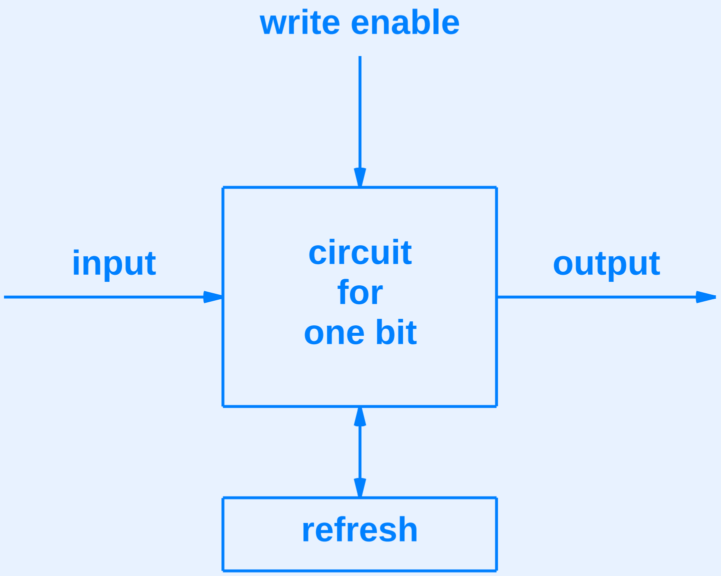
Dynamic RAM (DRAM)

- Alternative to SRAM
- Consumes less power
- Acts like a *capacitor* that stores an electrical charge

Making DRAM Work

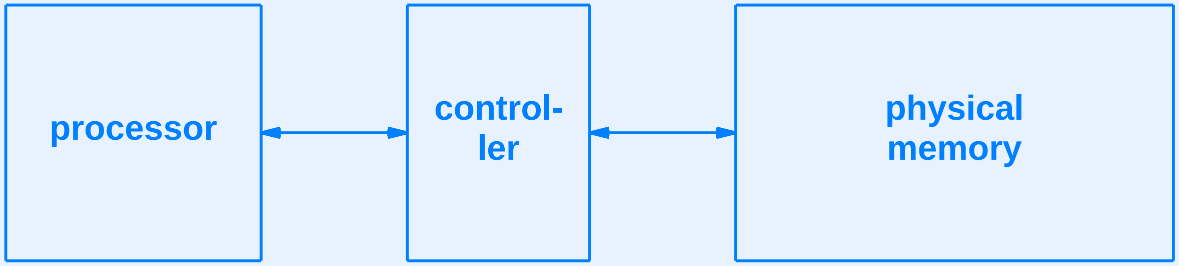
- Need extra hardware that operates independently
- Repeatedly steps through each location of DRAM
- Reads value from location in DRAM
- Writes value back into same location (recharges the memory bit)
- Extra hardware known as a *refresh circuit*

Illustration Of Bit In DRAM



Memory Organization

- Hardware unit connects processor to physical memory chips
- Called a *memory controller*

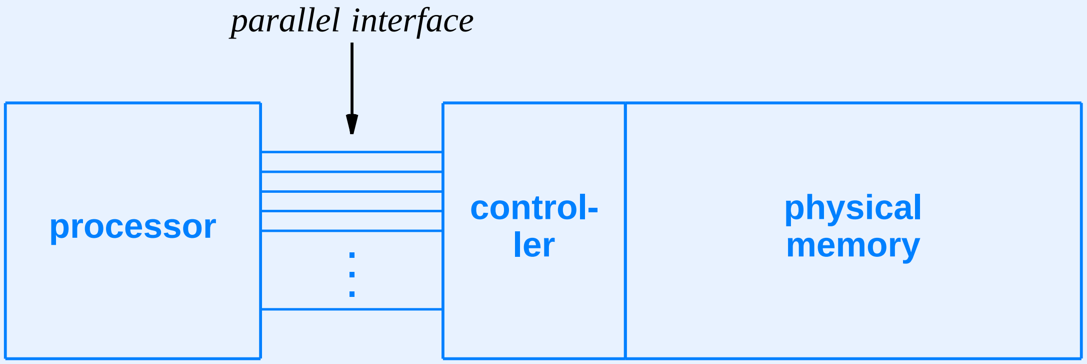


- Main point: because all memory requests go through the controller, the interface a processor "sees" can differ from the underlying hardware organization

Honoring A Memory Request

- Processor
  - Presents request to controller
  - Waits for response
- Controller
  - Translates request into signals for physical memory chips
  - Returns answer to processor immediately
  - Sends signals to reset physical memory for next request

Memory Organization



- Parallel interface used between computer and memory
- Called a *bus* (more later in the course)

Memory Transfer Size

- Amount of memory that can be transferred to computer simultaneously
- Determined by bus between computer and controller
- Example memory transfer sizes
  - 16 bits
  - 32 bits
  - 64 bits
- Important to programmers

Byte Addressing

- View of memory presented to processor
- Each byte of memory assigned an address
- Convenient for programmers
- Underlying memory can still use word addressing

Byte Alignment

- Refers to integer storage in memory
- In some architectures
  - Integer in memory must correspond to word in underlying physical memory
- In other architectures
  - Integer can be unaligned, but *fetch* and *store* operations are much slower

Memory Size And Address Space

- Size of address limits maximum memory
- Example: 32-bit address can represent

$$2^{32} = 4{,}294{,}967{,}296$$

unique addresses

- Known as *address space*
- Note: word addressing allows larger memory than byte addressing

Measures Of Physical Memory Size

Physical memory is organized into a set of M *words that each contain* N *bytes; to make controller hardware efficient,* M *and* N *are each chosen to be powers of two.*

- Consequence of the above: memory sizes expressed as powers of two, not powers of ten
  - Kilobyte defined to be 2^{10} bytes
  - Megabyte defined to be 2^{20} bytes

C Programming And Memory Addressability

- C has a heritage of both byte and word addressing
- Example of byte pointer declaration

```
char    *iptr;
```

- Example of integer pointer declaration

```
int     *iptr;
```

- If integer size is four bytes, iptr++ increments by four

Memory Dump

- Used for debugging
- Printable representation of bytes in memory
- Each line of output specifies memory address and bytes starting at that address

Example Memory Dump

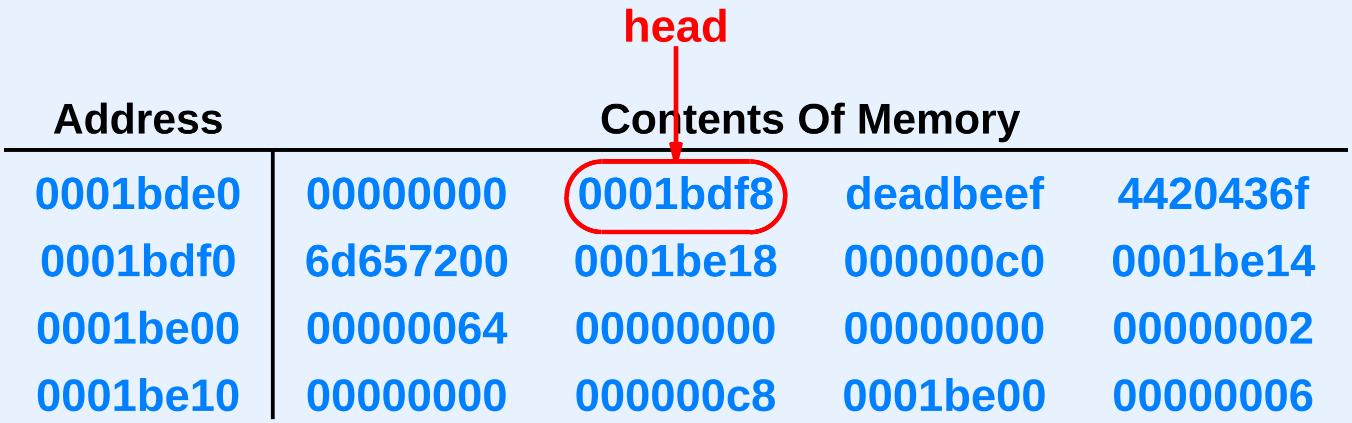
- Assume linked list in memory
- Head consists of pointer
- Each node has the following structure:

```
struct  node  {
        int count;
        struct node *next;
}
```

Example Memory Dump

```
 Address                  Contents Of Memory
---------------------------------------------------------
0001bde0 | 00000000  0001bdf8  deadbeef  4420436f
0001bdf0 | 6d657200  0001be18  000000c0  0001be14
0001be00 | 00000064  00000000  00000000  00000002
0001be10 | 00000000  000000c8  0001be00  00000006
```

Example Memory Dump

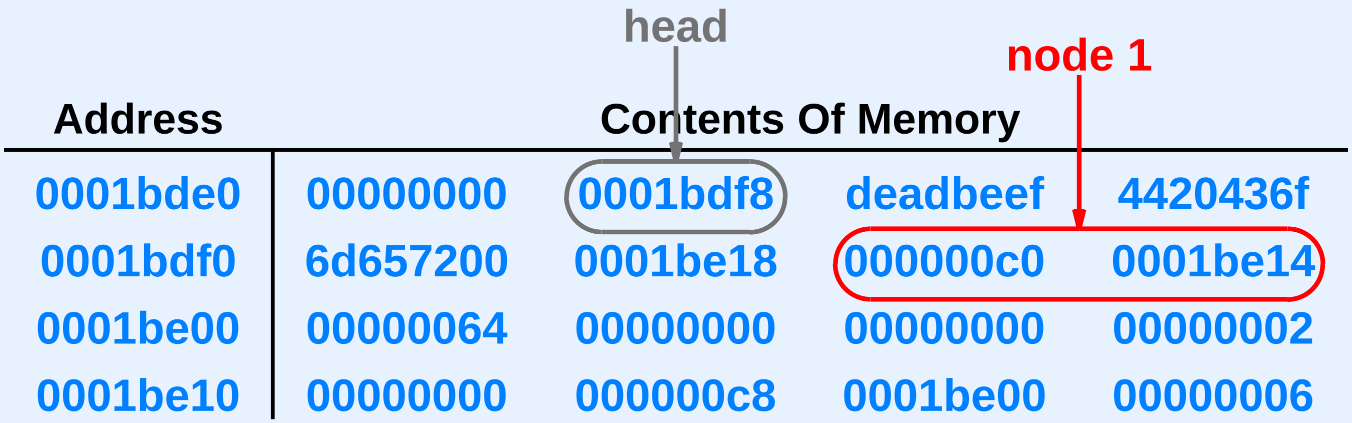


head

Address	Contents Of Memory			
0001bde0	00000000	0001bdf8	deadbeef	4420436f
0001bdf0	6d657200	0001be18	000000c0	0001be14
0001be00	00000064	00000000	00000000	00000002
0001be10	00000000	000000c8	0001be00	00000006

- Assume head located at address 0x0001bde4

Example Memory Dump



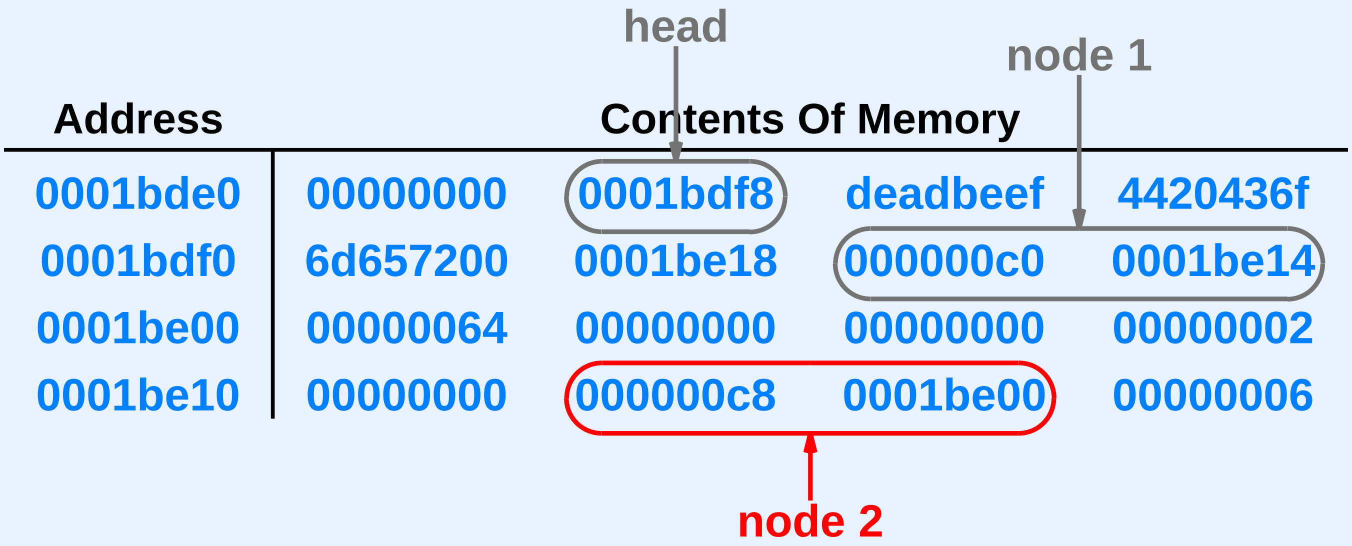
head

node 1

```
  Address                 Contents Of Memory
0001bde0 | 00000000   0001bdf8   deadbeef   4420436f
0001bdf0 | 6d657200   0001be18   000000c0   0001be14
0001be00 | 00000064   00000000   00000000   00000002
0001be10 | 00000000   000000c8   0001be00   00000006
```

- Assume head located at address 0x0001bde4
- First node at 0x0001bdf8 contains 192 (0xc0)

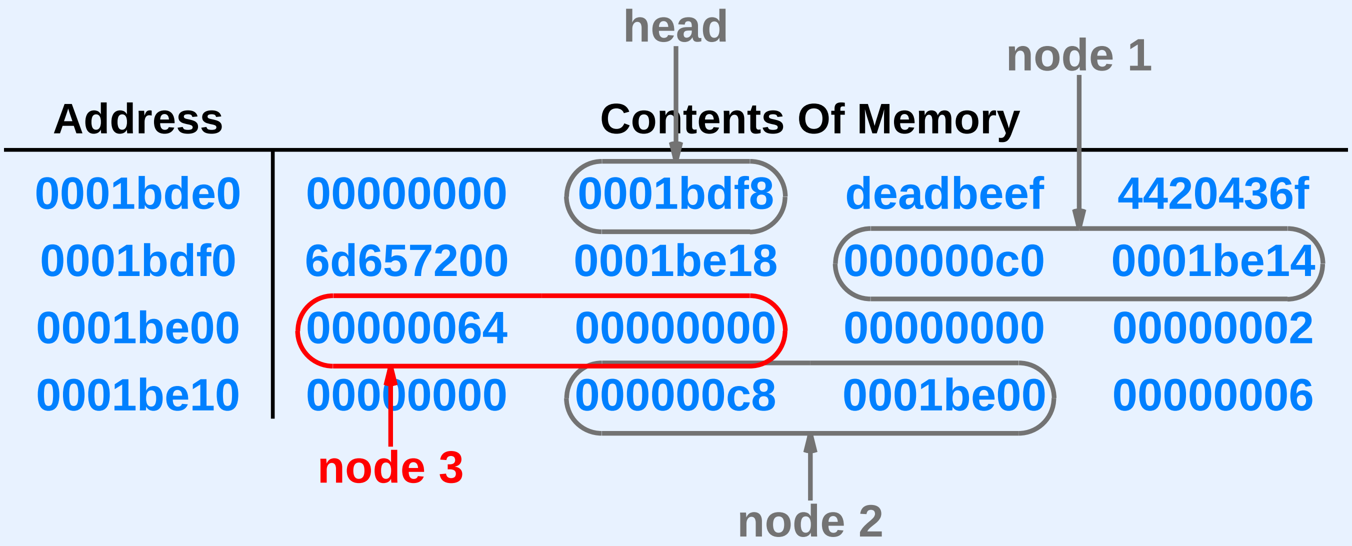
Example Memory Dump



```
Address      Contents Of Memory
0001bde0   00000000   0001bdf8   deadbeef   4420436f
0001bdf0   6d657200   0001be18   000000c0   0001be14
0001be00   00000064   00000000   00000000   00000002
0001be10   00000000   000000c8   0001be00   00000006
```

- Assume head located at address 0x0001bde4
- First node at 0x0001bdf8 contains 192 (0xc0)
- Second node at 0x0001be14 contains 200 (0xc8)

Example Memory Dump



```
Address        Contents Of Memory
0001bde0   00000000   0001bdf8   deadbeef   4420436f
0001bdf0   6d657200   0001be18   000000c0   0001be14
0001be00   00000064   00000000   00000000   00000002
0001be10   00000000   000000c8   0001be00   00000006
```

- Assume head located at address 0x0001bde4
- First node at 0x0001bdf8 contains 192 (0xc0)
- Second node at 0x0001be14 contains 200 (0xc8)
- Last node at 0x001be00 contains 100 (0x64)