How Linked List are Stored inside the Computer

- One of the few data structures that will become very clear in assembler is the linked list because you will need to manipulate the low level addresses to traverse the entire list
- A computer program needs information to do its work (solve some problem)

  A linked list is one of the many **data structures** that Computer Science has developed to manage/organize information

  The linked list is a **dynamic** structure, i.e., the number of elements in a linked list can **increase** and **decrease** in time

  (In contrast, the **array** data structure is **static**, number of elements stays constant)

- Structure of a linked list **element**
  - A linked list element **always** consists of 2 parts...
  - A **data** part consisting of one or more variables that hold the actual information you want to stored
  - A **linkage** part consisting of one or more **reference** variables that hold **addresses** of the **neighboring** linked list elements

  Symbolically:

```
Linked list element:

+-----------------------+
|                       |
|                       | <---- data part (variables that
|                       |       hold actually information
|                       |       that you want to store
+-----------------------+
|                       | <---- linkage part (reference variable
|                       |       containing address of neighboring
+-----------------------+       list elements
```

- Structure of a **linked list**
  - A linked list consists of zero or more linked list elements **chained** together by their **linkage** parts.
  - The special reference value **null** indicates "no more elements"
  - The **data** parts of the elements chained would contain the actual information stored

  Symbolically:

```
A simple linked list:

+--------+    -->+--------+    -->+--------+
```

```
| info1  |   /   | info2  |   /   | info3  |
|        |  /    |        |  /    |        |
+--------+ /     +--------+ /     +--------+
| ref1   |-      | ref2   |-      | null   |
+--------+       +--------+       +--------+
```

- The linked list elements are **linked** together by using **references** which are **addresses**

  Each linked list element is located **somewhere** in memory, and thus, each linked list element has a **starting** memory address

  The linkage **reference** variables contains the **address** of the next linked list element

  Example: the above linked list of 3 element would be linked together as follows:

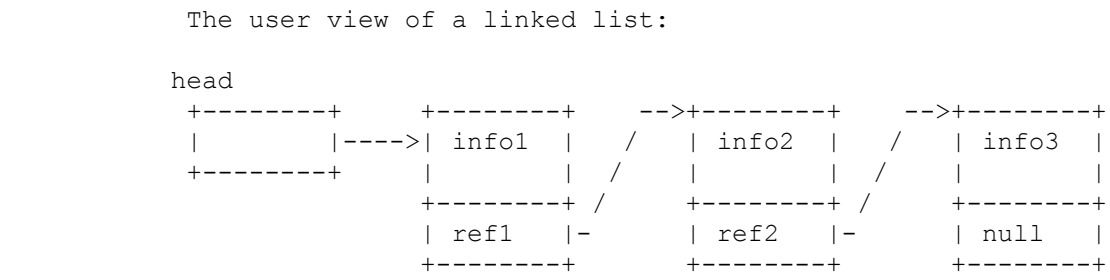
```
We have to make some assumptions to do the example.

The assumptions are:
       First element of linked is is located at address 8000
       Second element of linked is is located at address 10000
       Third element of linked is is located at address 9000

Memory:
       |               |
       +---------------+
 8000  |  info1        |  Linked list element 1
       +---------------+
       | ref1=10000    |-------------------------------------+
       +---------------+                                     |
       |               |                                     |
       +---------------+                                     |
         .......                                             |
         .......                                             |
       +---------------+                                     |
 9000  |  info3        |  Linked list element 3 <----+       |
       +---------------+                             |       |
       | ref3=0 (null) |                             |       |
       +---------------+                             |       |
         .......                                     |       |
         .......                                     |       |
       +---------------+                             |       |
10000  |  info2        |  Linked list element 2      | <-----+
       +---------------+                             |
       | ref2=9000     |-----------------------------+
       +---------------+
```

  - Notice that although I drew **arrows** from one linked list element to the next one, these arrows are just imaginary ones represented by the **address** value contained by the "ref" variables
  - (A reference in Computer Science is the same as a memory address !)

- To access the **entire** linked list (i.e., to access **all** elemenets in the linked list), you (only) need to know the **location** (**address**) of the **first** list element

  That is why you need an extra variable usually called **head** that contains a reference (**address**) of the first element in the linked list:

Symbolically:



This is what happens in reality: