Accessing Memory

- There is No memory - to- memory operations

  mov <address>, <register> # move data from memory to a register
  mov <register>, <address> # move data from register to memory

  <address> can be:

  - **An absolute address** (a number): 1000, or 2000, etc.

    Example: mov 10, %rax #load data from the address 10
    #this is different from mov $10,%rax
    (we almost never use this way usually because we don't know data location)

  - **Dereference a pointer** (pointers are always 64 bits, so use rdi, rcx, etc)
    Example: mov (%rcx), %rax #load data from the address saved in %rcx into %rax.
    Example: mov %rax,(%rdi) #store data from %rax into where %rdi points to.

  - **Using offset from pointer:** (used to access field of a struct)
    Example: mov 20(%rcx), %r8 #load data from the address saved in %rcx+20 into %rax.

  - **Index Addressing:** (used to access array elements)
    Example: mov (%rsi, %rcx,4), %rax #load data from the address saved
    # in %rsi + (4*%rcx) into %rax.

  - **Index Addressing with offset:** (used to access array elements in a struct)

    Example: mov 20(%rdi, %rcx,8), %rax #load data from the address saved
    # in 20+ %rdi + (8*%rcx) into %rax.

Example 1:
(Dereference a Pointer)

```c
#include <stdio.h>
#include <stdlib.h>
int getvalue(int* a );
int main(void) {
int* p=malloc(sizeof(int));
*p=5;
int v=getvalue(p);
printf("%d\n",v);
return 0;
}
```

```
  .globl _getvalue
_getvalue:
 push %rbp
 mov %rsp, %rbp
 mov (%rdi), %rax
 pop %rbp
 ret
```

Example 2:

(Access elements in a struct)

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct person{
   char name[6];
   int id;
   int salary;
};

int getid(struct person* );
int main(){
   struct person* p;
   p=malloc(sizeof(struct person));
   p->id=10;
   strcpy(p->name,"hasan");
   p->salary=2000;

   printf("%d\n",getid(p));

}
```

```asm
 .globl _getid
_getid:
 push %rbp
 mov %rsp, %rbp
 mov 8(%rdi), %rax
 pop %rbp
 ret
```

Example 3:

(Access elements in an array)

```c
#include<stdio.h>

int getElementAt(int arr[], int loc);
int main(){
   int a[5]={10,20,30,40,50};
   printf("%d\n",getElementAt(a, 2000));
}
```

```asm
   .globl _getElementAt
_getElementAt:
   push %rbp
   mov %rsp, %rbp

   mov (%rdi,%rsi,4),%rax

   pop %rbp
   ret
```

Example 4:

(Access elements in an array in a struct)

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct employee{
   int id;
   int payments[10];
   int salary;
};

int getPaymentAt(struct employee* p, int
payment_loc);
int main(){
   struct employee* e=malloc(sizeof(struct
employee));
   e->id=10;
   e->payments[0]=1000;
   e->payments[1]=2000;
   e->payments[2]=3000;
   e->payments[3]=4000;
   e->payments[4]=5000;
   e->salary=2500;
   printf("%d\n",getPaymentAt(e, 3));
}
```

```asm
   .globl _getPaymentAt
_getPaymentAt:
   push %rbp
   mov %rsp, %rbp

   mov 4(%rdi, %rsi,4),%rax

   pop %rbp
   ret
```

Example 5:

(Find the sum of array elements)

```c
#include<stdio.h>

int findSum(int arr[], int size);
int main(){
   int a[]={10,20,30,40,50};
   int s=findSum(a, 5);
   printf("%d\n",s);
}

/*
int findSum(int arr[], int size){
   int sum=0;
   for(int i=0; i<size;i++){
       sum+=arr[i];
   }
   return sum;
}
*/
```

```asm
   .globl _findSum

_findSum:
   push %rbp
   mov %rsp, %rbp

  # int sum=0;
  # for(int i=0; i<size;i++){
  #    sum+=arr[i];
  # }
  # a------> rdi
  # size--->  rsi
  # sum ---> rax
  # i -----> rcx
   mov $0, %rax
   mov $0, %rcx
Top:
   cmp %rsi, %rcx
   jge Done
   add (%rdi, %rcx, 4),%rax
   inc %rcx
   jmp Top
Done:
   pop %rbp
   ret
```