Design Document

OS Assignment 2

Paste your code corresponding to `push_back`

```
static void push_back(struct thread *t)
{
     struct thread *temp=ready_list;
     if(ready_list==NULL){
          ready_list=t;
          ready_list->next=NULL;
          ready_list->prev=NULL;
          return;
     }
     while (temp->next!=NULL){
          temp=temp->next;
     }
     temp->next=t;
     t->prev=temp;
     t->next=NULL;

}
```

Paste your code corresponding to `pop_front`.

```
static struct thread *pop_front()
{

     struct thread *toRet=ready_list;
```

```
	ready_list=ready_list->next;
	if(ready_list!=NULL)
		ready_list->prev=NULL;

	return toRet;

}
```

Paste your code corresponding to `create_thread`. If you are calling functions that are defined by you in `create_thread`, paste the code of them as well.

```
void create_thread(func_t func, void *param)
{
	unsigned *stack = malloc(4096);
	stack+=1024;
	unsigned *esp=stack;

	struct thread *new_thread = malloc(sizeof(struct thread));

	//Putting arguements in stack
	*esp=(unsigned)param;
	esp-=sizeof(unsigned)/4;
	//Putting fake return address
	*esp=(unsigned)0;
	esp-=sizeof(unsigned)/4;
	//Pushing function
	*esp=(unsigned)func;
	esp-=sizeof(unsigned)/4;
	//Putting ebx
	*esp=(unsigned)0;
	esp-=sizeof(unsigned)/4;
```

```
    //Putting edi
    *esp=(unsigned)0;
    esp-=sizeof(unsigned)/4;
    //Putting edi
    *esp=(unsigned)0;
    esp-=sizeof(unsigned)/4;
    //Putting ebp
    *esp=(unsigned)0;



    new_thread->esp=esp;
    new_thread->next=NULL;
    new_thread->prev=NULL;

    push_back(new_thread);
}
```

Dump the output of the“make test’’

```
gcc -Werror -m32 -O3 -c thread.c -g
gcc -m32 -c context.s -g
gcc -O3 -m32 thread.o context.o -o app app.c -g
./app 1024
starting main thread: num_threads: 1024
main thread exiting: counter:30768239300147200
./app 1024 1
starting main thread: num_threads: 1024
bar1: (nil)
bar2: (nil)
bar1: 0x1
main thread exiting: counter:0
```

Suggest a strategy to free struct thread and the stack corresponding to the thread that has exited

In thread exit function:

- get the base of the stack using the esp pointer of the struct thread. (By making the last 12 bits 0 of the address). Assumption: we use mymalloc() instead of malloc().
- Free the stack and current thread.

OR,

- Change the struct thread data structure and add another field to save the base of the stack while creating a new thread.
- Free the stack in thread_exit() function using the saved base pointer in thread struct thread.