MTE 241 Project 2

Jong Sha Han

Objectives:

Implement memory allocation and deallocation strategy using linked-list data structures and half-fit algorithm.

Summary

Half fit algorithm is a dynamic memory allocation/deallocation strategy considered suitable for real-time operating systems. When a request for a memory allocation comes in, a block is taken from the bin where the request is guaranteed to fit while a bit-vector is used to record which bins are empty. During deallocation, the freed block is coalesced with the adjacent blocks that are available. This prevents very small fragments from being left over (less internal fragmentations).

Use of bit-vectors makes it possible to immediately determine if the request can be satisfied without walking any lists. It allows the bucket the memory will come from to be determined without having to access the

array itself, and access the data at the address (two additional reads and a null check).

Initially, the entire managed memory space will be one block in the largest bucket. This block will then be divided based on the size of user requests. A header is used to manage the relationship between blocks. One doubly-linked list structure is used to track all the blocks in a bucket. A second doubly-linked list structure is used to manage adjacent blocks of memory.

Key Functions Implemented

```
void  half_init(void)
```

This function initializes a block header and a bucket header. Since doubly linked-list data structure is used, set prev and next points to null with is_alloc flag set 0 (free).

```
void *half_alloc(U32 size)
```

This is a memory allocation function which takes an integer as an argument and returns a void pointer to a block of memory of the appropriate size. A request for n bytes will access a block from the bucket corresponding to the least power of two, that is greater or equal to the requested amount. The address to be returned for memory points to the

start of the memory. Since a header takes up 4 bytes at the start of the block, start_index + 4 bytes will be provided.

```
void  half_free(void * address)
```

This is a de-allocation function that takes a block of memory and re-integrates it back into the memory pool. If the freed memory block is adjacent to other currently free memory blocks, it is then merged with them. The combined block is re-integrated into the correct bucket in the memory pool. The void pointer passed as an argument will provide the start address. The possible cases for coalescing are where left or right is free, or both are free.

```
void coalesce(block_header_t* left,block_header_t* right)
```

This function coalesces left and right currently available (free) blocks. It pops the right (left) bucket from the stack and pushes the coalesced bucket into the stack. Block header is updated accordingly.