Project 1: β-allocator

Computation structures

October 9, 2018

Memory allocation

- **Static allocation**: size must be known at compile-time, handled by compilers in high-level languages. In β-assembly:

```
| static allocation on the stack
.macro ALLOCATE(N) ADDC(sp, N*4, sp)
.macro DEALLOCATE(N) SUBC(sp, N*4, sp)
```

```
| static allocation at next byte position
.macro STORAGE(NWORDS)   . = .+(4*NWORDS)
```

- **Dynamic allocation**: size can be unknown at compile-time, programmers controls the lifespan of the allocated memory. Not possible natively in β-assembly.

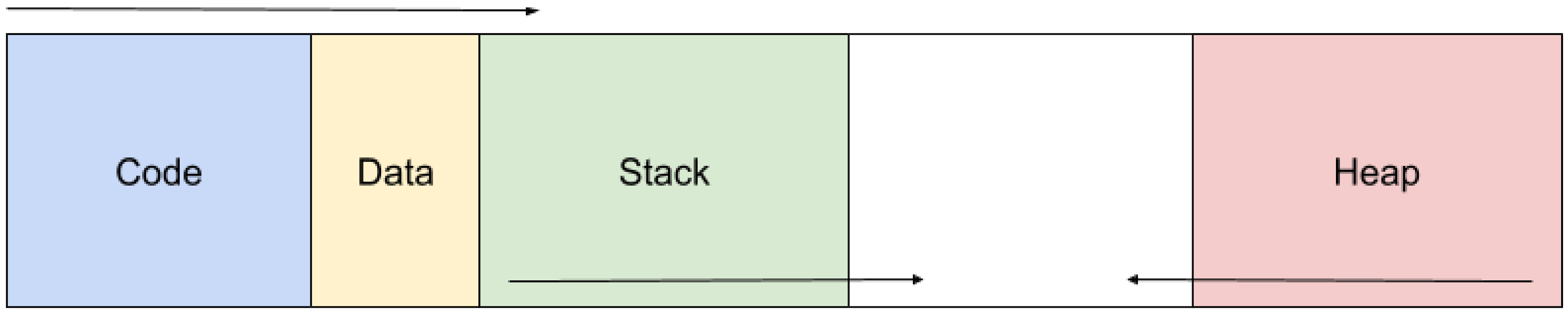
β-allocator

Your task is to implement the following interface in β-assembly:

- `int* malloc(int n)`: allocates n words and returns a pointer to the rst word of the allocated space.
- `void free(int* p)`: frees the allocated memory space pointed by p

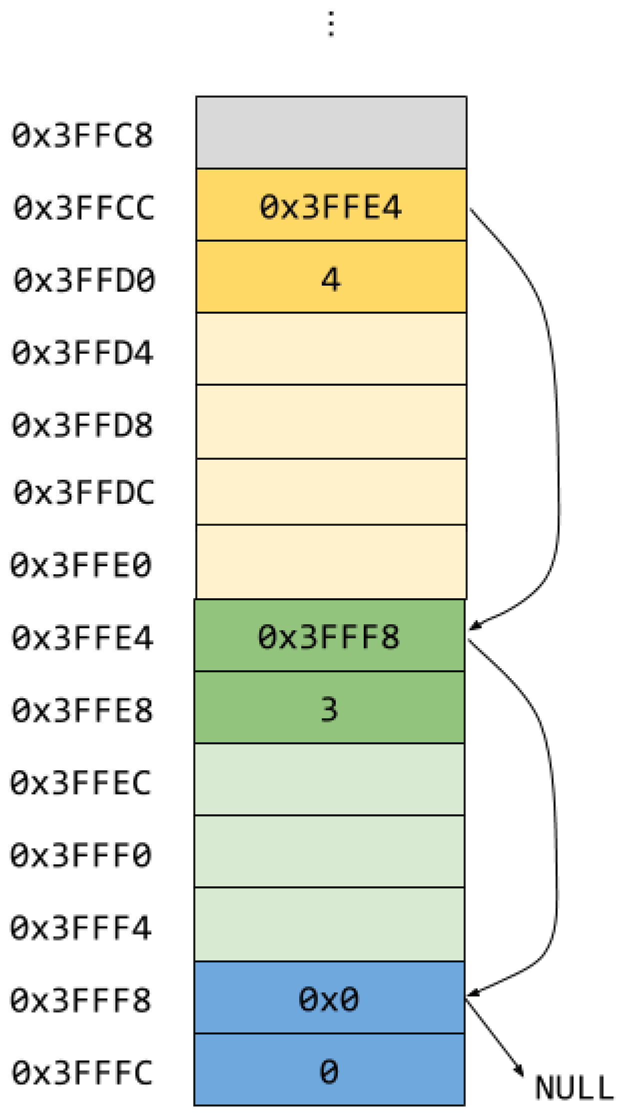
β-allocator - heap

- we will implement a memory **heap** where will be stored the allocated memory.
- it will grows from higher addresses towards lower ones

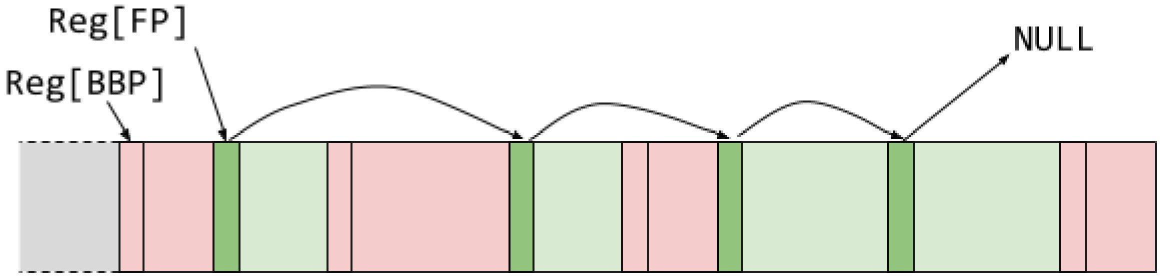


β-allocator - memory blocks

- The heap contains **memory blocks**
- A memory block represents a chunk memory that can be allocated for the program
- A memory block consists in:
  - A two-words header:
    1. `next`: a pointer to the header of a another block (32bits)
    2. `size`: the size of the memory chunk stored in the block (32bits)
  - A chunk of memory of containing `size` words



β-allocator – free list



- When calling `malloc`, one have to **find a block** of which the chunk is **large enough** to store the allocated array.
- One need to identify free blocks easily ⇒ let us use a linked list of free blocks
- Each free block `next` pointer **points to the first free block** following it in memory
- Two new dedicated registers:
  - **BPP** (R26): **Base Block Pointer**, first block of the heap)
  - **FP** (R25): **Free Pointer**, first free block of the heap

β-allocator – implementation

- **Initialization**: heap is initialized with one empty block
- **Malloc**:
  - Look for the first block large enough in the free list
  - If there is no such block, create a new one at the beginning of the heap
- **Free**:
  - Insert the freed block at the right position in the list
  - Merge block with its neighbours if they are contiguous (to avoid fragmentation)

- More details in the project statement

β-allocator – files

You are provided with:

- **`malloc.c`** : a C implementation of the interface. You can use it as basis for your assembly implementation.
- **`beta.usam`** : definition of the β-assembly. Check this file to see which macro you can use.
- **`main.asm`** : a file that initializes the heap and calls `malloc` and `free`.
- **`malloc.asm`**: a skeleton to fill with your implementation

You must submit a zip file containing:

- **malloc.asm** : your implementation of the β-allocator
- **(optional) report.pdf** : if you think you need more than the comments to explain some parts of your code, you can write those explanations in a short report (maximum two pages).
- **Submitting other files will result in a penalty** (or submitting a folder inside the zip)

Practical details

- **By groups of two students**
  - You can find teammates on the submission platform
- **Deadline**: **November 5, 2018 at 23:59**

Submission:

- Submit your archive on the Montefiore submission platform
- You can only submit 10 times !
- When submitted, be sure that your implementation passes the **automated tests** !