Introdução à Ciência de Computação I
- Exercício 10 -
MEMÓRIA RAM E SUAS SEGMENTAÇÕES

Aluno: Hélio Nogueira Cardoso; **N°USP**: 10310227; **Data**: Junho de 2021

1. Explicar, diferenciar e exemplificar os funcionamentos de ambos segmentos de memória Stack e Heap.

A memória RAM (Random-Access Memory ou Memória de Acesso Aleatório) é uma forma de memória de computador que pode ser lida e alterada para armazenar dados e códigos de máquina. Ela é dividida em segmentos: Stack e Heap.

Stack

- A Stack é uma estrutura de dados linear;
- A memória é alocada em blocos sequenciais (contíguos e contínuos);
- A Memória Stack é alocada e desalocada automaticamente utilizando instruções do próprio compilador;
- É menos custoso para o programador manter bom funcionamento da Stack;
- É fácil de implementar;
- Não é flexível. É estática, fixa em tamanho;
- Se todos os blocos são ocupados, a memória também se esgota;
- Toma-se menos tempo para acessar os elementos da Stack.

Heap

- A Heap é uma estrutura de dados hierárquica;
- A memória é alocada de forma aleatória;
- A Memória Heap é alocada e desalocada manualmente pelo programador (Em C, é administrada através de chamadas de sistema para funções como malloc, calloc, free, delete etc);
- É mais custoso para o programador manter bom funcionamento da Heap;
- É mais difícil de implementar uma estrutura Heap;
- Toma-se mais tempo para acessar os elementos da Heap;
- A desvantagem da Heap é a fragmentação da memória;
- Redimensionamento é possível na Heap, portanto memória não é desperdiçada.

Tabela de Comparação

Comparação	Stack	Heap
Alocação	Sequencial	Aleatória e hierárquica
Alocação/Desalocação	Automático	Manual
Custo	Baixo	Alto
Implementação	Fácil	Difícil
Invocação	Complexidade O(N)	Complexidade O(1)
Desvantagem	Curto em tamanho	Fragmentação da memória
Localidade de Referência	Excelente	Adequada
Flexibilidade	Fixo em tamanho e inflexível	Redimensionamento possível
Tempo de Acesso	Mais rápido	Mais lento

2. Como é organizado o armazenamento na Stack? O armazenamento na Heap é feito da mesma forma?

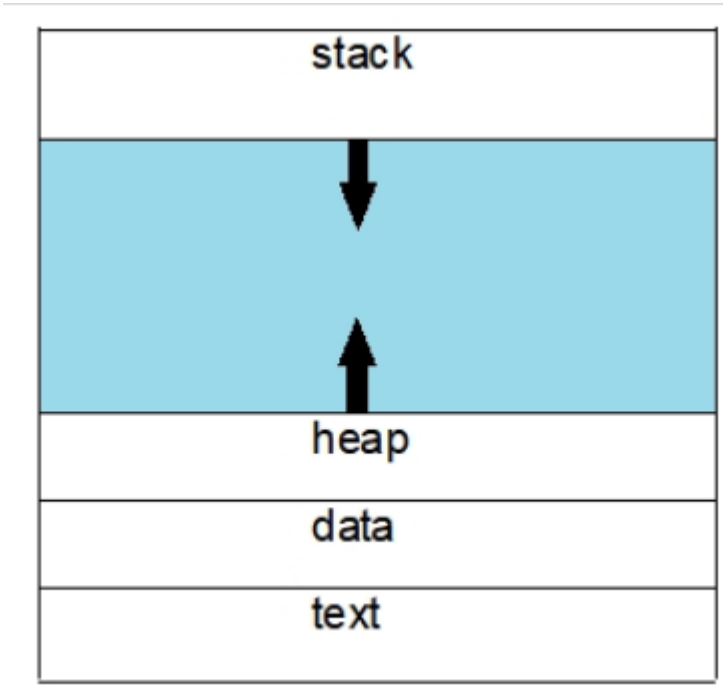
Uma boa tradução para Stack é Pilha. A memória Stack pertece ao programa que criamos. Ela é organizada de forma linear e é alocada e desalocada automaticamente. Cada método/função possui sua própria pilha. Estes metódos são também, por assim dizer, empilhados. Dessa forma, por exemplo, um programa em C começa pela função principal main(), a qual poderá possuir suas próprias variáveis declaradas na Stack. Quando uma função é chamada dentro da main(), como uma função de soma(), por exemplo, este metódo é "colocado em cima da pilha", de maneira que a main() para de ser executada. Agora, a função soma() tem sua própria pilha com as variáveis de seu escopo. Assim que a função soma() termina de ser executada, ela é "retirada da pilha", de modo que todas as variáveis de sua pilha são automaticamente desalocadas e a main() volta a ser executada "no topo da pilha". Ao fim do programa, todas as variáveis da Stack que pertecíam a ele são também automaticamente desalocadas. As variáveis são sempre declaradas estáticas na Stack, com tamanho pré-fixado.

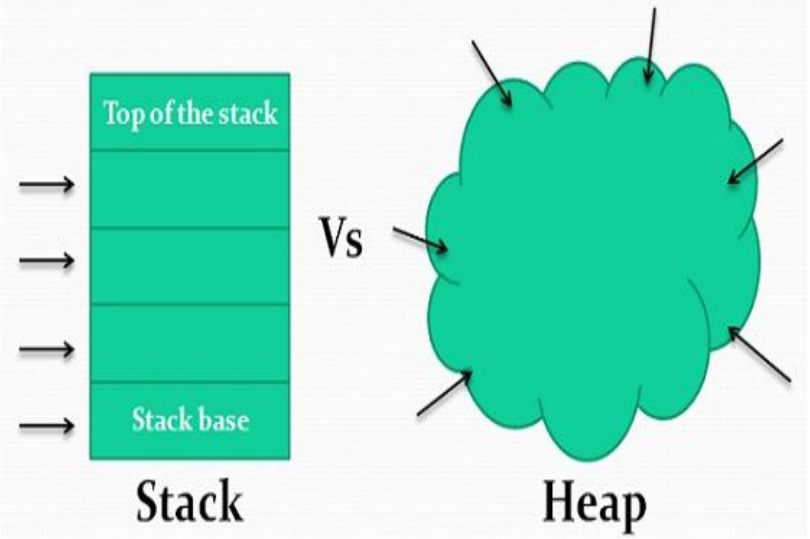
Uma boa tradução para Heap é Monte. Ela não pertence de fato ao nosso programa, mas é sim um Monte de memória compartilhada ao longo do computador. O nosso programa pode pedir por memória na Heap, armazenando ponteiros para as devidas posições requisitadas. Isso nos permite redimensimensionar os dados, o que torna a memória Heap flexível e boa para guardar dados grandes ou que não sabemos bem a priori o tamanho. Como consequência negativa, é necessário que, após a completa utilização, liberemos as memórias alocadas manualmente para que outros programas as possam acessar. A memória Heap é organizada de forma hierárquica ("estrutura de árvore") e os dados são alocados em endereços aleatórios.

Memória num processo

Os endereços da Stack, na maioria das arquiteturas, cresce a partir de endereços altos em direção a endereços mais baixos.

A memória Heap compreende um grande espaço em endereços mais baixos cuja alocação é feita de forma aleatória, não-sequencial e hierárquica.





3. As duas segmentações da memória diferem muito em tamanho (ou capacidade de armazenamento)?

Sim. A Stack, em especial, é uma memória limitada em tamanho. Já a Heap possui muito mais capacidade de armazenamento.

4. Em que situações é aconselhável que utilizemos a memória Stack? E a memória Heap?

Não se pode alocar dados na memória Stack antes de saber o seu tamanho (na hora de alocar na execução, não necessariamente no momento de compilação). Portanto, por ser de manutenção mais simples e menos custosa, devemos preferir usar a Stack quando sabemos de antemão o tamanho do nosso dado ou quando este tamanho será sabido no momento da execução em que ele é alocado e esse tamanho não é muito grande, pois a Stack tem tamanho limitado.

Quando precisamos alocar dados grandes, potencialmente grandes ou que não podemos determinar o tamanho no momento da alocação, provavelmente devemos utilizar a memória Heap, que é onde podemos fazer alocação dinâmica.

5. Existem prejuízos em abusar da memória Stack?

Sim, particularmente porque ela é limitada. Existe um tipo de erro muito famoso chamado “Stack Overflow”, que ocorre quando há um estouro da pilha. Ou seja, uma vez que a Stack é estática, se forem empilhados mais blocos do ela tem capacidade, isso levará a um erro.

Referências

- https://www.tutorialspoint.com/difference-between-stack-and-heap
- https://en.wikipedia.org/wiki/Random-access_memory
- https://www.tutorialspoint.com/how-does-a-process-look-like-in-memory
- https://www.guru99.com/stack-vs-heap.html
- https://techdifferences.com/difference-between-stack-and-heap.html