**CS 367 - Recitation #14 Allocators and the Heap,**

**Virtual Memory and Caching**

**Name Adilet Kuroda G#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Group Member Name**: **Javi Talavera**

**Group Member Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Today's Goals:** We want to get comfortable with dynamic memory management and the heap:

* common approaches to representing free/allocated blocks, e.g. with headers and implicit lists
* selecting blocks to allocate (like first fit, next fit, and best fit)
* coalescing strategies

We’re also going to get some practice with basic exceptional control flows and looking at processes.

**Work in groups of 2-3 students.** Every group will turn in what they've got to Blackboard.

Once again, there's probably more here than you might finish in 50 minutes, but it's great test preparation.

**Grading is based on participation.** Get as much done as you can. You will also be given feedback in the form of a 'score' (1-3) and possibly some comments. This doesn’t affect your grade – it is solely for feedback. A score of 3 means everything looks great.   A score of two indicates some minor problems.  And a score of one indicates that there were some major issues. If you get a 1, don't panic - go see your prof or a GTA to get more extensive feedback.

**Allocation Strategies**

We compare three strategies in this section:

* **first fit**, starting at the beginning of the heap and advancing through blocks to find the first large-enough free block
* **next fit**, starting at the last-allocated block and advancing through blocks to find the first large-enough free block
* **best fit**, which searches for the closest-to-perfect size block available

In our lecture we have learned two approaches to tracking free blocks:

* implicit lists, where a block's size is enough information to find the following block (free or not), so we can scan the heap.
  + **Assume implicit list for all questions on this recitation, unless otherwise stated.**
* explicit lists, where pointers to a next and previous free block are also stored, allowing us to only consider free blocks (but headers/footers still allow us to traverse/identify all blocks, free or not)

**Allocating Blocks**

Using various indicated block selection strategies, be the allocator and handle all the allocate/free requests. **Draw out your work and upload the drawings directly to Blackboard.**

* Each square represents a word, and the requests indicate the number of blocks to be allocated/freed, including any headers/footers (you do not need to add extra squares for bookkeeping and size words)
* Padding will be done to Double-Word (size will be an even number of words)
* Alignment will be done to Double-Word (starting address will be even)
* If a malloc request is not possible, write the strategy that fails in front of that request.
* Assume that best-fit breaks ties with the leftmost space that exactly fits

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **First fit** | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P1 | P1 | P1 | P1 |  |  |  |  | P5 | P5 | P5 |  | P3 | P3 | P3 | P3 | P3 | P3 |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| **Next fit** | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P1 | P1 | P1 | P1 |  |  |  |  |  |  |  |  | P3 | P3 | P3 | P3 | P3 | P3 |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| **Best fit** | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P1 | P1 | P1 | P1 |  |  |  |  | P5 | P5 | P5 |  | P3 | P3 | P3 | P3 | P3 | P3 |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

1. p1 = malloc(4)  
   p2 = malloc(8)  
   p3 = malloc(6)  
   free(p2)  
   p4 = malloc(4)  
   p5 = malloc(3)  
   free(p4)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **First fit** | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 | P3 | P3 | P3 | P4 | P4 | P4 | P4 | P4 | P4 | P4 | P4 | P4 | P4 | P2 | P2 | P2 | P2 | P2 | P2 |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| **Next fit** | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | P2 | P2 | P2 | P2 | P2 | P2 | P3 | P3 | P3 | P3 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| **Best fit** | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 | P3 | P3 | P3 | P4 | P4 | P4 | P4 | P4 | P4 | P4 | P4 | P4 | P4 | P2 | P2 | P2 | P2 | P2 | P2 |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

1. p1 = malloc(14)  
   p2 = malloc(6)  
   free(p1)  
   p3 = malloc(4)  
   p4 = malloc(10)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **First fit** | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 | P3 | P3 | P3 |  |  |  |  | P5 | P5 | P5 | P5 |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| **Next fit** | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | P3 | P3 | P3 | P3 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| **Best fit** | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 | P3 | P3 | P3 |  |  |  |  | P5 | P5 | P5 | P5 |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

1. p1 = malloc(10)  
   p2 = malloc(10)  
   free(p1)  
   p3 = malloc(4)  
   free(p2)  
   p4 = malloc(4)  
   p5 = malloc(3)  
   free(p4)

**Page Table and VA --> PA calculations**

|  |  |  |
| --- | --- | --- |
| VPN | Valid? | PPN |
| 000 | 1 | 000 1101 |
| 001 | 1 | 000 1111 |
| 010 | 0 | 011 0011 |
| 011 | 1 | 101 1000 |
| 100 | 1 | 100 1111 |
| 101 | 0 | 100 1101 |
| 110 | 1 | 100 1011 |
| 111 | 1 | 001 0101 |

Assume a process uses 8-bit virtual addresses where 3 bits are the virtual page number. The page table is given to the right.

1. How many of the process's pages are currently in memory?

6

1. How many bits does a physical address have here?

12

1. What is the size of a page here? (# addressable bytes in it)

8

1. Calculate physical addresses (PA's) from the following virtual addresses, or indicate that a page fault would occur. Answer in hexadecimal.
2. VA = 0x14 = 0001 0100

0x 1B4

1. VA = 0x58 = 0101 1000

Page fault

1. VA = 0x9F = 1001 1111

0x9FF

|  |
| --- |
| 100 1111 |

**Translation Lookaside Buffers (TLBs)**

The information about the system and address sizes are at the top of the next page.

* **You will need to do a little deduction to determine the TLB Tag and Index, the Cache Tag and Index, as well as the Page Numbers and Offsets.**

|  |  |  |  |
| --- | --- | --- | --- |
| TLB | | | |
| Idx | Tag | PPN | Valid |
| 0 | 07 | 02 | 1 |
| 1 | 03 | 2D | 1 |
| 2 | 00 | – | 0 |
| 3 | 07 | – | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| L1 Cache | | | | | | |
| Idx | Tag | Valid | Blk 0 | Blk 1 | Blk 2 | Blk 3 |
| 0 | 19 | 1 | 99 | 11 | 23 | 11 |
| 1 | 15 | 0 | – | – | – | – |
| 2 | 1B | 1 | 00 | 02 | 04 | 08 |
| 3 | 0D | 1 | 23 | A5 | 67 | 89 |
| 4 | 32 | 1 | 43 | 6D | 8F | 09 |
| 5 | 0D | 1 | 36 | 72 | F0 | 1D |
| 6 | 31 | 0 | – | – | – | – |
| 7 | 16 | 1 | 11 | C2 | DF | 03 |
| 8 | 24 | 1 | 3A | 00 | 51 | 89 |
| 9 | 2D | 0 | – | – | – | – |
| A | 2D | 1 | 93 | 15 | DA | 3B |
| B | 0B | 0 | – | – | – | – |
| C | 12 | 0 | – | – | – | – |
| D | 16 | 1 | 04 | 96 | 34 | 15 |
| E | 13 | 1 | 83 | 77 | 1B | D3 |
| F | 14 | 0 | – | – | – | – |

|  |  |  |
| --- | --- | --- |
| Page Table | | |
| VPN | PPN | Valid |
| 00 | 28 | 1 |
| 01 | – | 0 |
| 02 | 33 | 1 |
| 03 | 02 | 1 |
| 04 | – | 0 |
| 05 | 16 | 1 |
| 06 | – | 0 |
| 07 | – | 0 |
| 08 | 13 | 1 |
| 09 | 17 | 1 |
| 0A | 09 | 1 |
| 0B | – | 0 |
| 0C | – | 0 |
| 0D | 2D | 1 |
| 0E | 11 | 1 |
| 0F | 0D | 1 |

|  |  |
| --- | --- |
| **Terminology Used:** | |
| **VPN** | Virtual Page Number |
| **PPN** | Physical Page Number |
| **Idx** | Index |
| **Blk X** | Block, Offset X |

Perform the entire byte lookup, from VA to PA to byte for each of these Three translations.

* Since our VA's are not a multiple of 4 bits, make sure to start conversions from Hex to Binary from the right, converting four bits at a time.
* We're addressing a single byte (your answers are single bytes if found).
* Use some scratch paper to do the work and enter your answers here!

0000 0011 0110 1010

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **VA = 0x36A** |  | **VA = 0x733** |  | **VA = 0x024** |
| **Part I: Virtual Address** | | | | |
| TLB Tag: 00000011 |  | TLB Tag: |  | TLB Tag: |
| TLB Index: 01 | TLB Index: |  | TLB Index: |
| VPO: 101010 | VPO: |  | VPO: |
| **Part II: Translation** | | | | |
| TLB Hit?yes |  | TLB Hit? |  | TLB Hit? |
| PPN Result: 0010 1101 | PPN Result: |  | PPN Result: |
| **Part III: Physical Address** | | | | |
| Cache Tag: 0000 1011 |  | Cache Tag: |  | Cache Tag: |
| Cache Index: 1010 | Cache Index: |  | Cache Index: |
| Cache Offset:10 | Cache Offset: |  | Cache Offset: |
| **Part IV: Data Fetch** | | | | |
| Cache Hit?miss |  | Cache Hit? |  | Cache Hit? |
| Data Result: N/a | Data Result: |  | Data Result: |