Worksheet 9
CS/ECE 354 - Spring 2016

Due: April 27th 2016 (Wednesday) in class

1. A computer has the following characteristics:
   a. 4GB Virtual Address Space
   b. 2GB Physical Address Space
   c. Page Size (P) = 4KB

   How many bits are required to specify the following?

   If you do not have enough information to answer the question, say “Not Enough Info”.

   Virtual Page Number (VPN): \( 82 - VPO = 20 \)

   Physical Page Number (PPN): \( 31 - PPO = 19 \)

   Virtual Page Offset (VPO): \( 12 \)

   Physical Page Offset (VPO): \( PPO = 12 \)

   Maximum number of Page Table Entries (PTE) per process: \( 2^{20} \)

   Number of page table entries (PTE) in TLB: \( NEI \)

   Swap space (in bytes): \( NEI \)

2. Determine the minimum block size for each of the following combinations of alignment requirements and block formats.

   Assumptions:
   1. Explicit free list
   2. 4-byte prev and next pointers in each free block
   3. zero-sized payloads are not allowed

   for allocated blocks //
Need to address 4 GB.

VA → 32 bits.

Need to address 2 GB.

PA → 31 bits.

\[ P = 4 \text{ KB} \]

\[ VPO = 10 \] bits \]

\[ VPO = PPO = \log_2 (4 \text{ KB}) \]

\[ = \log_2 (4096) = 12 \text{ bits} \]

\[ 1 \text{ KB} = 1024 \text{ bytes} \]

\[ 1 \text{ KB} = 2^{10} \text{ bytes} \]

\[ 1 \text{ KB} = 1000 \text{ bytes} \]
No space for payload.

Why? Because when it is freed, we get 8 bytes of space for the payload.
4. Headers and footers are stored in 4-byte words.
5. Block size = sizeof(header) + sizeof(payload) + sizeof(padding) + sizeof(footer) + sizeof(pointers)

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Allocated Block</th>
<th>Free Block</th>
<th>Minimum block size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Word</td>
<td>Header and Footer</td>
<td>Header and Footer</td>
<td>16</td>
</tr>
<tr>
<td>Single Word</td>
<td>Header but no footer</td>
<td>Header and Footer</td>
<td></td>
</tr>
<tr>
<td>Double Word</td>
<td>Header and Footer</td>
<td>Header and Footer</td>
<td></td>
</tr>
<tr>
<td>Double Word</td>
<td>Header but no footer</td>
<td>Header and Footer</td>
<td></td>
</tr>
</tbody>
</table>

3. Determine the **maximum block size** for each of the following combinations of alignment requirements and header sizes.

**Assumptions:**
- 1. Implicit free list
- 2. Block size = sizeof(header) + sizeof(payload) + sizeof(padding)

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Header Size</th>
<th>Maximum block size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Word</td>
<td>1 byte</td>
<td>252</td>
</tr>
<tr>
<td>Single Word</td>
<td>2 bytes</td>
<td></td>
</tr>
<tr>
<td>Double Word</td>
<td>1 byte</td>
<td>248</td>
</tr>
<tr>
<td>Double Word</td>
<td>2 bytes</td>
<td>248</td>
</tr>
</tbody>
</table>

Single word → payload should start at (\(\times 4\)) → side effect → each block's size is a multiple of 4.
When this allocated block is freed, will we have enough space for the prev/next pointers?

Allocated Block

- Header: 4
- Footer: 4
- Total: 16
4. Headers and footers are stored in 4-byte words.
5. Block size = sizeof (header) + sizeof (payload) + sizeof (padding) + sizeof (footer) + sizeof (pointers)

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Allocated Block</th>
<th>Free Block</th>
<th>Minimum block size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Word</td>
<td>Header and Footer</td>
<td>Header and Footer</td>
<td>16</td>
</tr>
<tr>
<td>Single Word</td>
<td>Header but no footer</td>
<td>Header and Footer</td>
<td></td>
</tr>
<tr>
<td>Double Word</td>
<td>Header and Footer</td>
<td>Header and Footer</td>
<td></td>
</tr>
<tr>
<td>Double Word</td>
<td>Header but no footer</td>
<td>Header and Footer</td>
<td></td>
</tr>
</tbody>
</table>

3. Determine the **maximum block size** for each of the following combinations of alignment requirements and header sizes.

**Assumptions:**
1. Implicit free list
2. Block size = sizeof (header) + sizeof (payload) + sizeof (padding)

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Header Size</th>
<th>Maximum block size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Word</td>
<td>1 byte</td>
<td>252</td>
</tr>
<tr>
<td>Single Word</td>
<td>2 bytes</td>
<td></td>
</tr>
<tr>
<td>Double Word</td>
<td>1 byte</td>
<td>248</td>
</tr>
<tr>
<td>Double Word</td>
<td>2 bytes</td>
<td></td>
</tr>
</tbody>
</table>

\[ 111111111 \rightarrow 255! \]

- Single word \rightarrow payload should start at \((\times 4)\)
- Side effect \rightarrow each block's size is a multiple of 4.
8 = 1000
12 = 1100
16 = 10000

8 = 1000
12 = 1100
16 = 10000

11111 1100 \rightarrow 252

11111 1000 \rightarrow 248
Last class

1. Signals

2. Examples

3. Handling

---

CORRECTION

CTRL + Z \(\not\rightarrow\) SIGSTOP

CTRL + Z \(\rightarrow\) SIGTSTOP

(Stop signal not from terminal)

(Stop signal is from the terminal)

Print vs write
We talked about how signal() is not standard.

So we use POSIX

\[ \text{Sigaction}() \]

\[ \rightarrow \text{portable!} \]
\[ \rightarrow \text{implemented in unix, linux, OSX} \]

What to do when a sys call is interrupted by a signal?

How to block a signal?

Hope you book notes!
int sigaction (int signum,
                      struct sigaction *act,
                      struct sigaction *oldact)

set this
based on our
needs

Ignore this
for now.