In Page Map (aka, Page Table, PTbl)
One entry per virtual page
Resident bit (R, also called “valid” bit) = 1 if page in physical memory
DIRTY bit = 1 if page contents have been changed since loaded into physical memory
Arithmetic

\[2^p\] \quad \text{bytes per physical page}

\[(v+p)\] \quad \text{bits in virtual address}

\[(m+p)\] \quad \text{bits in physical address}

\[2^v\] \quad \text{number of virtual pages}

\[2^m\] \quad \text{number of physical pages}

\[(m+2)2^v\] \quad \text{bits in page map}

Operating system issues:

“OS Kernel” is a special, privileged process that oversees the other processes and handles real I/O devices

Each process has its own Process Control Block (PCB), which encapsulates its state

Scheduler() switches among user processes

\[
\text{struct MState} \{
\quad \text{int Regs}[31]; \quad /* saved state of user's registers */
\} \quad \text{User};
\]

\[\text{int N} = 42; \quad /* number of processes to schedule */\]

\[\text{int Cur} = 0; \quad /* number of "active" process */\]

\[
\text{struct PCB} \{
\quad \text{struct MState State; \quad /* processor state */}
\quad \text{Context PageMap; \quad /* VM map for process */}
\quad \text{int DPYNum; \quad /* console/keyboard number */}
\quad \text{ProcTbl[N]; \quad /* one per process */}
\} \quad \text{Scheduler}() \{
\quad \text{ProcTbl[Cur].State} = \text{User}; \quad /* save current user state */
\quad \text{Cur} = (\text{Cur} + 1) \% \text{N}; \quad /* increment modulo N */
\quad \text{User} = \text{ProcTbl[Cur].State; \quad /* make another process the current one */}
\}
\]

Fine print:

Quiz is closed-book, no calculators; covers Building the Beta, Caches, Virtual Memory, OS issues (Virtual Machines) -- up to L18 (Virtual Machines)/R18 (this recitation)

Practice, practice, practice:

Follow “Previous terms” link from http://6004.csail.mit.edu, pick a semester (the more recent, the better), click on the “Announcements” page for the semester, and find the PDF for Quiz 4 and 5 solutions. **NOTE:** We covered material in different order this year, skipped over some subjects, and focused more on others. Do not worry about set associative caches; instead, worry about SVC’s and questions like Problem 3 of today's tutorial.

Another perspective on the material – Margaret Chong’s Handbook:

Follow “Handouts” link from http://6004.csail.mit.edu, click on handbook link near the bottom of the page.

Handouts

Make sure you understand as much as you can of the Unpipelined Beta diagram and Control Logic chart (provided on page 1).
Virtual Memory, revisited

Problem 1, Part G:
The table to the left shows the first 8 entries in the page map. Recall that the valid bit is 1 if the page is resident in physical memory and 0 if the page is on disk or hasn’t been allocated.

If there are 1024 ($2^{10}$) bytes per page, what is the physical address corresponding to the decimal virtual address 3956?

<table>
<thead>
<tr>
<th>Virtual page</th>
<th>Valid bit</th>
<th>Physical page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

OS issues

Problem 2:

- = VEC_RESET
  BR(I_Reset) | on Reset (start-up)

- = VEC_ILI
  BR(I_IlOp)  | on Illegal Instruction  
  (eg SVC)

- = VEC_CLK
  BR(I_Clk) | On clock interrupt

- = VEC_KBD
  BR(I_Kbd) | on Keyboard interrupt

- = VEC_MOUSE
  BR(I_BadInt) | on mouse interrupt

... 
I_Reset:
CMOVE(P0Stack, SP)
CMOVE(P0Start, XP)
JMP(XP)

Problem 1:

A

ReadKey_h() {
  int kdbnum = ProcTbl[Cur].DPYNum;
  while (BufferEmpty(kdbnum)) {
    /* busy wait loop */
  }
  User.Regs[0] = ReadInputBuffer(kdbnum);
}

C

ReadKey_h() {
  int kdbnum = ProcTbl[Cur].DPYNum;
  if (BufferEmpty(kdbnum)) {
    Scheduler();
  } else
    User.Regs[0] = ReadInputBuffer(kdbnum);
}

D

ReadKey_h() {
  int kdbnum = ProcTbl[Cur].DPYNum;
  if (BufferEmpty(kdbnum)) {
    Scheduler();
  } else
    User.Regs[0] = ReadInputBuffer(kdbnum);  
}