in-class exam.
Close books, close notes.
Calculator OK.

This exam is given under the Georgia Tech Honor Code System. You must observe and sign the Honor Pledge: “I have neither given nor received aid on this exam.” Your print name and signature below signifies your compliance with this honor code.

Name (Print): SOLUTION

Signature: _________________________________________________

1. _____________ (25 pts)
2. _____________ (35 pts)
3. _____________ (20 pts)
4. _____________ (20 pts)

Total (100 pts) ____________
1. (25%) Q&A. Keep your answer *concise* within **one or two sentences**.

1.1. Name the five possible states of a process during a process’ life.

   New, Ready, Running, Waiting, Terminate

1.2. Which of the following components can be shared by multiple threads that belong to the same process?
   (a) Stack pointer; (b) Global data variables; (c) Code; (d) Register file; (e) Program counter

   (b) Global data variables   and   (c) Code

1.3. Is a TLB absolutely necessary for successful address translations? Why?

   No. TLB is simply a small cache to capture locality for page table entries.

1.4. When does a **page fault** take place?

   When a memory page is not present in memory (i.e. no translation is found in page table), a page fault is triggered to bring the corresponding page into memory.

1.5. Explain how can “starvation” occur when priority scheduling is applied. Also explain the solution to resolve it.

   Low priority processes are never scheduled for CPU time. Aging can be used to promote a lower priority process to higher priority over time.
2. Virtual Memory (35%). Assume an **Inverted Page Table (8-entry IPT)** is used by a 32-bit OS. The memory page size is **2MB**. The complete IPT content is shown below. The Physical Page Number (PPN) starts from **0** to **7** from the top of the table. There are three active processes, P1 (PID=1), P2 (PID=2) and P3 (PID=3) running in the system and the IPT holds the translation for the entire physical memory. Answer the following questions.

<table>
<thead>
<tr>
<th>Valid</th>
<th>Process ID (PID)</th>
<th>Virtual Page Number (VPN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0x3fe</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0x001</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0x1ad</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0x7fd</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0x3fe</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0x2bf</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0x7fd</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0x0bf</td>
</tr>
</tbody>
</table>

2.1. (5%) Based on the size of the Inverted Page Table above, what is the size (in MB) of the physical memory?

**Since there are only 8 entries in the IPT, thus there are 8 physical frames.**

\[ 8 \times 2\text{MB} = 16\text{MB} \] **physical memory.**

2.2. (10%) To which “physical address” does the “virtual address” **0x7fdd8f64** of P2 map? Please derive and write down the complete address in **Hex** value. (If there is no valid mapping, please answer “page fault”)

\[ 2\text{MB} \rightarrow 21 \text{ bits page offset} \]
\[ 32 - 21 = 11 \text{ bits virtual page number (VPN)} \]
\[ 7fdd8f64 = 0111~1111~1101~1101~1000~1111~0110~0100 \]
\[ \text{VPN} = 0111~1111~110 = 0x3fe \rightarrow \text{the 5th entry in the IPT} \rightarrow \text{PPN} = 100 \]
\[ \text{Physical address} = 1001~1101~1000~1111~0110~0100 = 0x9d8f64 \]
2.3. (10%) To which “virtual address”, of “which process”, does the physical address 0x78e968 map? Please derive and write down the complete address in Hex value. If you cannot find a valid mapping, please answer “address not found”.

0x78e968 = 0111 1000 1110 1001 0110 1000
PPN = 011 = 3 → the 4th entry which has a valid entry with VPN=0x7fd for PID=3
Virtual address = 0x7fd || 1 1000 1110 1001 0110 1000
Virtual address = 1111 1111 1011 1000 1110 1001 0110 1000
Virtual address = 0xffb8e968 of Process P3

2.4. (10%) Now the OS writer decides to use an index-based linear page table for each process. How big (in bits or bytes) the total page tables are needed for these active processes? Assume you have 2 extra bits (valid and dirty) in addition to the PPN for each page table entry.

From 2.2, we know the VPN has 11 bits. In other words, the total number of entries of an index-based linear page table will be $2^{11} = 2048$ possible translations.

Since there are only 8 pages in physical memory, thus PPN = 3 bits.
Each page table entry = 3 + 2 (v and d bits) = 5 bits
5 * 2048 * 3 processes = 15*2K bits = 30Kbits = 3.75KB
3. Process fork (20%). Given the following code, fill in the correct (compound) conditions in the question marks (???) so that all the new children and grandchildren processes will enter the if-clause while only the original master process enters the else-clause.

```c
#include <stdio.h>
#include <unistd.h>

main()
{
    int ret_pid, ret_ppid;
    int mypid;

    ret_pid = fork();
    ret_ppid = fork();
    mypid = getpid();

    if (???) {
        /* All Children/Grandchildren execute here */
        fprintf(stdout, "Child : mypid is %d\n", mypid);
        exit(0);
    } else {
        /* The Master comes here */
        fprintf(stdout, "Master: mypid is %d\n", mypid);
        wait(NULL);
        exit(0);
    }
}
```

**Version 1:**

```
if (ret_pid == 0 || ret_ppid == 0)
```

or

**Version 2:**

```
if ( ! (ret_pid != 0 || ret_ppid != 0) )
```
4. CPU Scheduling (20%). Given the following processes with their arrival times in ready queue and their respective predicted CPU burst times, please show your schedule in Gantt Chart (showing timeline) and calculate the average waiting time when use Pre-emptive Shortest Job Scheduling (or Shortest-Remaining Time First).

<table>
<thead>
<tr>
<th>Process ID</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

Wait time for A = 10
Wait time for B = 0
Wait time for C = 10 – 4 = 6
Wait time for D = 1
Wait time for E = 1
Average wait time = \((10+6+1+1)/5 = 3.6\) time units