### Input Details:

### $AMJ000100030001 Job ID: 0001, Total Time limit: 0003, Total line limit: 0001

### GD10PD10H Instructions to be executed

### $DTA Start of data card

### Hello World Data

### $END 0001 End Job 0001

### Basic Concepts

**Virtual Memory:** This is the memory address space that a process sees and interacts with. It allows each process to have its own address space, providing an abstraction layer over the actual physical memory.

**Real Memory:** This is the actual physical memory (RAM) where the data is stored. The operating system maps virtual memory addresses to real memory addresses.

**Memory Structure:** Memory is divided into 30 blocks, each containing 10 words, where each word is 4 bytes. This totals to 300 words. Page tables are used to manage the mapping between virtual and real memory addresses.

### Components of the Simulator

### ****PCB (Process Control Block):**** Holds information about the job, including Job ID, Total Time Limit (TTL), Total Line Limit (TLL), Total Time Count (TTC), and Line Limit Count (LLC).

**OS Class:**

* **Memory (M):** 2D array representing the simulated memory.
* **Instruction Register (IR):** Holds the current instruction being executed.
* **Instruction Counter (IC):** Points to the next instruction to be executed.
* **General Purpose Register (R):** Used for various operations during execution.
* **System Interrupts (SI, PI, TI):** Handles various interrupts like system, program, and timer interrupts.
* **Buffers and Flags:** Used for I/O operations and memory management.

### Function Descriptions and Execution Flow

**INIT:**

* Initializes all variables and memory to their default states.
* Allocates a block for the page table.

**Load:**

* Reads the input file and loads the job instructions into memory.
* Handles control cards ($AMJ, $DTA, $END) to manage the start and end of jobs and data loading.

**Execute:**

* Executes the instructions loaded into memory.
* Handles different operations like GD (Get Data), PD (Put Data), LR (Load Register), SR (Store Register), CR (Compare Register), and BT (Branch on Toggle).

**MOS (Master Mode):**

* Manages interrupts and system calls, directing the flow to appropriate functions like read, write, and terminate.

**Add\_map:**

* Maps virtual addresses to real addresses, handling page faults by allocating new blocks as needed.

**Read and Write:**

* Read: Reads data from the input file into memory.
* Write: Writes data from memory to the output file.

**Terminate:**

* Handles the termination of jobs, ensuring proper cleanup and output formatting.

### Detailed Sequence of Execution

**Initialization (INIT):**

* Memory, registers, and counters are initialized to zero or default values.
* A block is allocated for the page table.

**Loading Job Instructions (Load):**

* Reads the input file line by line.
* Control cards ($AMJ, $DTA, $END) are processed to manage job start, data loading, and job end.
* Instructions are loaded into memory, and the page table is updated accordingly.

**Executing Instructions (Execute):**

* Fetches the instruction from memory using the instruction counter (IC).
* Decodes and executes the instruction.
* Handles various operations (GD, PD, LR, SR, CR, BT) with corresponding actions and checks.
* Manages interrupts and system calls through MOS.

**Memory Management (Add\_map):**

* Translates virtual addresses to real addresses.
* Handles page faults by allocating new memory blocks and updating the page table.

**I/O Operations (Read/Write):**

* Read: Fetches data from the input file and stores it in memory.
* Write: Outputs data from memory to the output file, formatting it as needed.

**Termination (Terminate):**

* Cleans up after a job, ensuring proper output formatting and resetting states for the next job.

### Error Handling

**Page Faults:**

* Occur when a required page is not in memory. The system allocates a new block and updates the page table.

**Interrupts:**

* **System Interrupts (SI):** Managed through MOS, directing to appropriate functions (read, write, terminate).
* **Program Interrupts (PI):** Handle errors like invalid operations or operand errors.
* **Timer Interrupts (TI):** Ensure jobs do not exceed their time limits.

### Example Inputs and Errors

1. **Out of Data**

**Description:** This error occurs when a GD (Get Data) instruction tries to read more data than is available in the input.

$AMJ000100070003

GD10GD20PD10H

$DTA

HELLO WORLD!

$END0001

**Cause:** The input specifies two GD instructions, but the data segment only contains one line of data.

1. **Line Limit Exceeded**

**Description:** This error occurs when the number of lines written to the output exceeds the Total Line Limit (TLL).

$AMJ000100070000

GD10PD10H

$DTA

HELLO WORLD!

$END0001

**Cause:** The input $AMJ001000070000 specifies a TLL of 0, but the program attempts to write 1 line due to PD (Put Data) instructions.

1. **Time Limit Exceeded**

**Description:** This error occurs when the Total Time Count (TTC) exceeds the Total Time Limit (TTL).

$AMJ000100020003

GD10PD10H

$DTA

HELLO WORLD!

$END0001

**Cause:** The input $AMJ000100020003 specifies a TTL of 2, but the program execution takes more time (time required is 3 (2 for GD and 1 for PD) due to multiple instructions and possible page faults.

1. **Operation Code Error**

**Description:** This error occurs when an invalid operation code is encountered in the instruction set.

$AMJ000100070003

GX10PD10H

$DTA

HELLO WORLD!

$END0001

**Cause:** The input includes an invalid instruction GX10, which is not recognized by the system.

1. **Operand Error**

**Description:** This error occurs when the address specified in an instruction is not valid.

$AMJ000100070003

GD100PD10H

$DTA

HELLO WORLD!

$END0001

**Cause:** The input $AMJ000100070003 includes an instruction GD100, which specifies a 3-digit address, leading to an operand error.

1. **Invalid Page Fault**

**Description:** This error occurs when a page fault cannot be resolved because no free memory blocks are available.

**Cause:** This error cannot be simulated with the given inputs, as the system allocates memory dynamically and handles page faults by assigning new blocks.