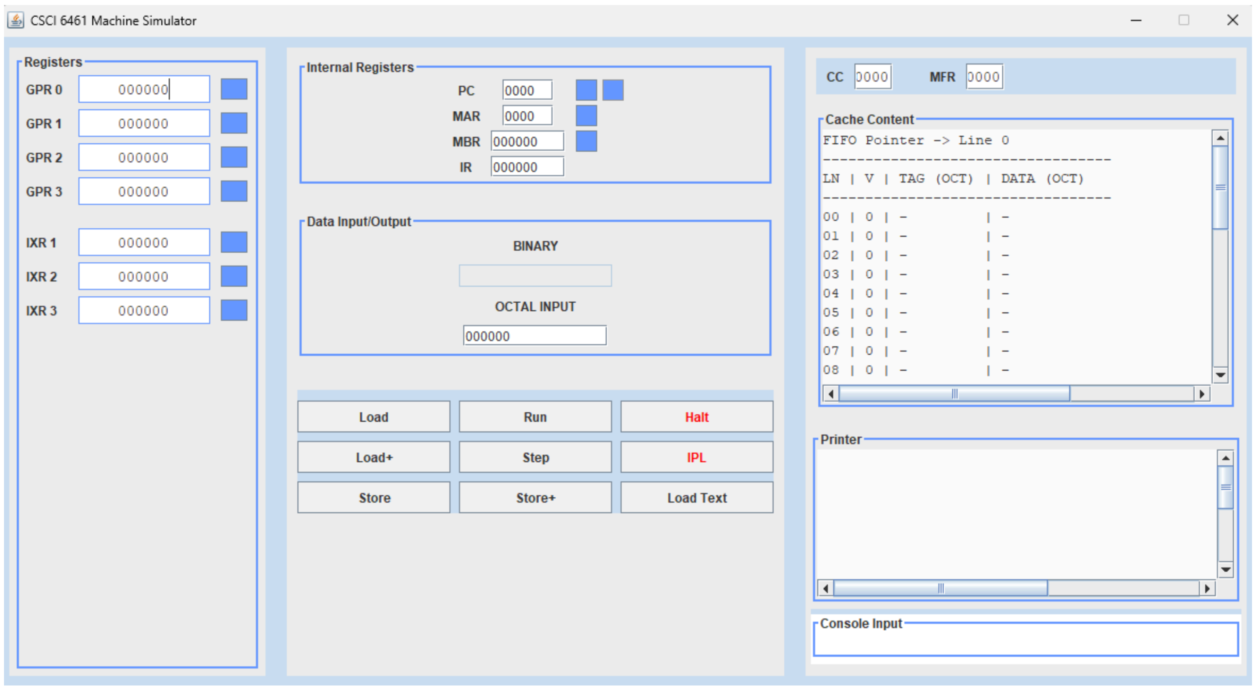
<u>CSCI 6461 Machine Simulator Documentation - Team 12</u>

https://github.com/jayparmar16/csci_6461_Group12/tree/Project3

1. The Console Layout

The simulator interface is divided into three main columns: Registers (left), Operations (center), and I/O (right).



GPR 0 000000
GPR 1 000000
GPR 2 000000
GPR 3 000000
IXR 1 000000
IXR 2 000000
IXR 3 000000

PC 0000
MAR 0000
MBR 000000
IR 000000

OCTAL INPUT 000000

CC 0000 MFR 0000

```
FIFO Pointer -> Line 0
-----------------------------------
LN | V | TAG (OCT) | DATA (OCT)
-----------------------------------
00 | 0 | -         | -
01 | 0 | -         | -
02 | 0 | -         | -
03 | 0 | -         | -
04 | 0 | -         | -
05 | 0 | -         | -
06 | 0 | -         | -
07 | 0 | -         | -
08 | 0 | -         | -
```

2. Left Panel: Registers

This panel displays the primary user-accessible registers.

GPR 0-3 (General Purpose Registers): Four 16-bit registers for general data manipulation.
IXR 1-3 (Index Registers): Three 16-bit registers used for indexed addressing calculations.
Register Buttons: The small blue buttons next to each GPR and IXR are non-functional placeholders. To load a value into these registers, you must use instructions like LDR or LDX from a program.

3. Center Panel: Operations & Internal State

This panel contains internal CPU registers, data input fields, and the main control buttons.

Internal Registers:

PC (Program Counter): A 12-bit register that holds the address of the next instruction to be executed.
MAR (Memory Address Register): A 12-bit register that holds the address of the memory location to be accessed.
MBR (Memory Buffer Register): A 16-bit register that holds the data being transferred to or from memory.
IR (Instruction Register): A 16-bit register that holds the current instruction being executed.
Load Buttons: The small blue button next to each internal register allows you to load a value directly from the OCTAL INPUT field into that specific register.

Data Input/Output:

BINARY: A read-only field that displays the 16-bit binary equivalent of the value in the OCTAL INPUT field.
OCTAL INPUT: The primary field for manual data entry. All values for registers or memory operations are entered here as octal numbers.

Operation Buttons:

IPL (Initial Program Load): The most important button. Click this to open a file dialog and load a program file into the simulator's memory. This resets the machine state.
Run: Executes the loaded program continuously until a HLT instruction is encountered or the Halt button is pressed.
Step: Executes a single instruction at the address pointed to by the PC.
Halt: Stops a continuously running program.
Load / Store: Executes a memory load/store operation using the value in the OCTAL INPUT field as the memory address.
Load+ / Store+: Similar to Load/Store, but first increments the PC before performing the operation.

4. Right Panel: I/O & Status

This panel displays status information, memory content, and program output.

Status Registers:

CC (Condition Code): A 4-bit register that stores the status (e.g., overflow, underflow) of the last arithmetic operation.
MFR (Machine Fault Register): A 4-bit register used to identify the type of machine fault that has occurred.
Cache Content: A text area that displays the contents of the machine's memory (2048 words). The display is formatted as Address: Value. **(Not Implemented Yet)**
Printer: A text area that shows a detailed log of each instruction as it is executed. This is useful for debugging and tracing program flow.
Console Input: A field for providing input to a running program when required by an input instruction.

5. How to Operate the Simulator (Detailed Steps)

Follow these steps to load and run a program.

Step 1: Start the Simulator

Run the application. The simulator window will appear with all registers and memory initialized to zero.

Step 2: Load a Program via IPL

- Click the IPL button. It is highlighted in red for emphasis.
- A file chooser dialog will open. Navigate to and select your program file (e.g., load_file.txt).
- Click Open.
- The simulator will load the program into memory. The PC will automatically be set to the starting address specified in the program file (e.g., 0006).

Step 3: Run and Enter the Input

- **Set paragraph start:** In the `OCTAL INPUT` box, enter **`000100`** (this is the octal address `100`, where the paragraph buffer starts in the sample program).
- **Load into R0**: Click the small blue load button next to `GPR 0` (it loads the octal value into `GPR0`). This ensures the program's R0 points to the paragraph buffer.
- **Load your paragraph file:** Click `Load Text`, choose your `.txt` (e.g. `test_input.txt`); the GUI will load it at the address shown in `OCTAL INPUT` and set `GPR0`/`GPR1` appropriately.
- **Run:** Click `Run` (or `Step` to observe instruction flow). The printer area will show the paragraph and the `?` prompt.
- **Enter Input:** Enter word input in Console Input and press “Enter”
- **Output:** Output format would be as follows
  - If input found: *{Input Word} {Position of Word} {Position of Sentence}*
  - Else: NOTFOUND

Step 4: Halting and Resetting

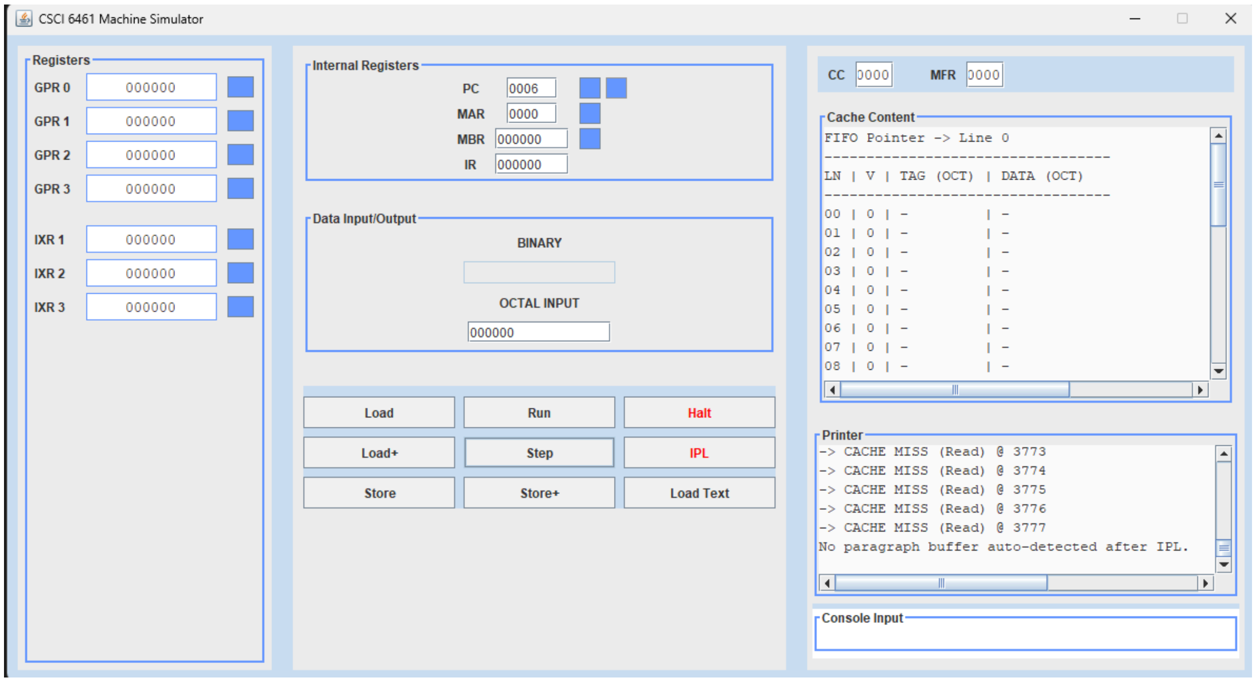
- To stop a program that is running continuously (e.g., if it's in an infinite loop), click the Halt button.
- To reset the machine and load a new program (or the same one again), simply go back to Step 2 and click the IPL button. The IPL process always resets the machine state before loading the new file.

Example Run

Example file contains the following text :

The sun rose quickly, birds chirped outside, and a breeze passed by as the phone rang once, she grabbed her coat, and they hurried away.

After IPL the load_file.txt :



Registers

GPR 0	000000
GPR 1	000000
GPR 2	000000
GPR 3	000000
IXR 1	000000
IXR 2	000000
IXR 3	000000

Internal Registers

PC	0006
MAR	0000
MBR	000000
IR	000000

Data Input/Output

BINARY

OCTAL INPUT: 000000

CC: 0000 MFR: 0000

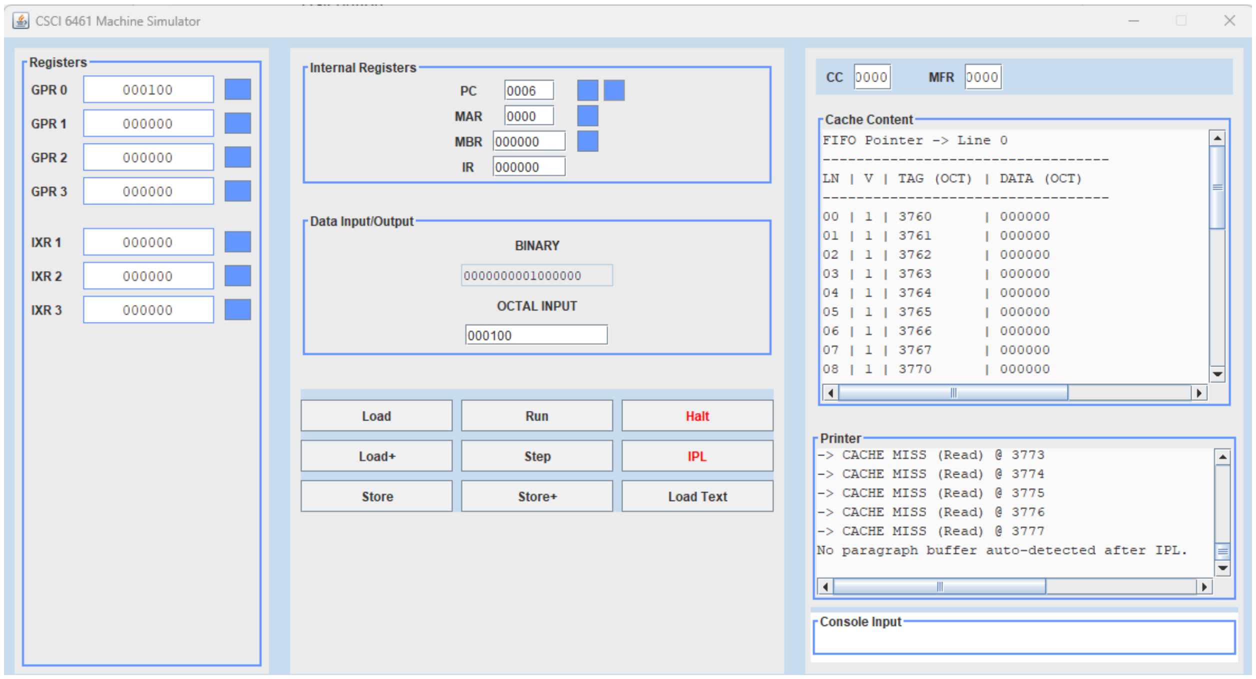
Cache Content

```
FIFO Pointer -> Line 0
----------------------------------
LN | V | TAG (OCT) | DATA (OCT)
----------------------------------
00 | 0 | -         | -
01 | 0 | -         | -
02 | 0 | -         | -
03 | 0 | -         | -
04 | 0 | -         | -
05 | 0 | -         | -
06 | 0 | -         | -
07 | 0 | -         | -
08 | 0 | -         | -
```

Printer

```
-> CACHE MISS (Read) @ 3773
-> CACHE MISS (Read) @ 3774
-> CACHE MISS (Read) @ 3775
-> CACHE MISS (Read) @ 3776
-> CACHE MISS (Read) @ 3777
No paragraph buffer auto-detected after IPL.
```

Inputting Octal Input as 000100 and clicking Blue Button for GPR 0:



Registers

GPR 0 000100
GPR 1 000000
GPR 2 000000
GPR 3 000000

IXR 1 000000
IXR 2 000000
IXR 3 000000

Internal Registers

PC 0006
MAR 0000
MBR 000000
IR 000000

Data Input/Output

BINARY 0000000001000000

OCTAL INPUT 000100

CC 0000 MFR 0000

Cache Content

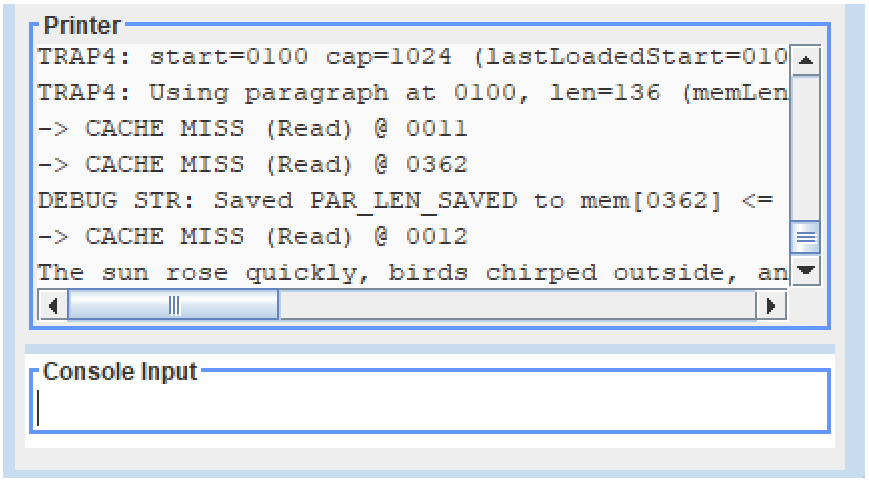
```
FIFO Pointer -> Line 0
-----------------------------------
LN | V | TAG (OCT) | DATA (OCT)
-----------------------------------
00 | 1 | 3760      | 000000
01 | 1 | 3761      | 000000
02 | 1 | 3762      | 000000
03 | 1 | 3763      | 000000
04 | 1 | 3764      | 000000
05 | 1 | 3765      | 000000
06 | 1 | 3766      | 000000
07 | 1 | 3767      | 000000
08 | 1 | 3770      | 000000
```

Printer

```
-> CACHE MISS (Read) @ 3773
-> CACHE MISS (Read) @ 3774
-> CACHE MISS (Read) @ 3775
-> CACHE MISS (Read) @ 3776
-> CACHE MISS (Read) @ 3777
No paragraph buffer auto-detected after IPL.
```

Console Input

Click Load Text and select the source file (.txt) and click Run.

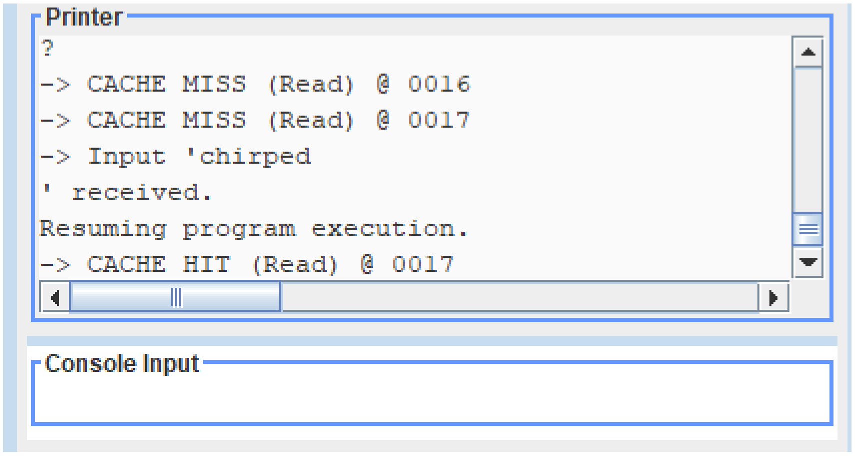


Printer

```
TRAP4: start=0100 cap=1024 (lastLoadedStart=010
TRAP4: Using paragraph at 0100, len=136 (memLen
-> CACHE MISS (Read) @ 0011
-> CACHE MISS (Read) @ 0362
DEBUG STR: Saved PAR_LEN_SAVED to mem[0362] <=
-> CACHE MISS (Read) @ 0012
The sun rose quickly, birds chirped outside, an
```

Console Input

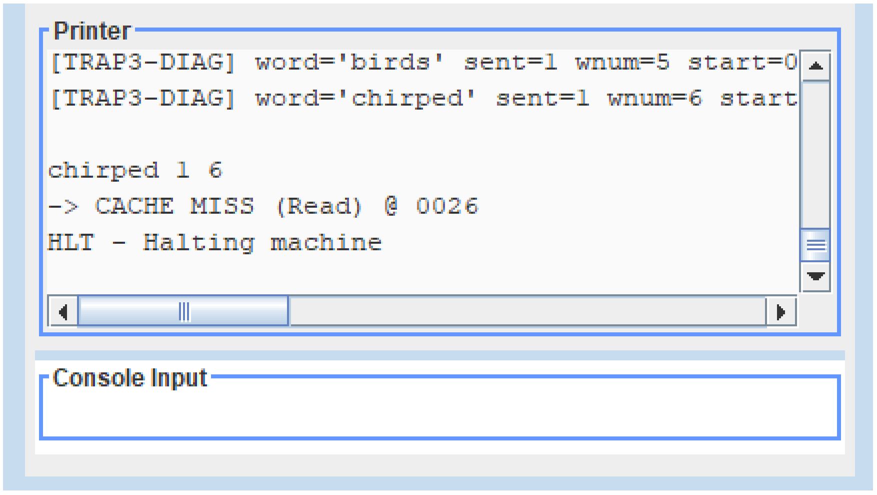
Enter Input “chirped” and hit Enter



Printer

```
?
-> CACHE MISS (Read) @ 0016
-> CACHE MISS (Read) @ 0017
-> Input 'chirped
' received.
Resuming program execution.
-> CACHE HIT (Read) @ 0017
```

Console Input



Printer

```
[TRAP3-DIAG] word='birds' sent=1 wnum=5 start=0
[TRAP3-DIAG] word='chirped' sent=1 wnum=6 start

chirped 1 6
-> CACHE MISS (Read) @ 0026
HLT - Halting machine
```

Console Input