CSCI 6461 Front Panel Simulator

User Guide for the Offline Browser Simulator

Use this guide with: C6461_front_panel_pc_fault_fixed.html

Purpose: The simulator models the current 16-bit CSCI 6461 machine in a web browser. It provides front-panel register loading, 2,048 words of memory, single-step and continuous execution, console input/output, TRAP processing, and machine-fault handling.

Critical notation rule: PC and MAR are shown as four-digit octal addresses. GPR, IXR, MBR, and IR are shown as six-digit octal words. MFR and CC are four-bit binary displays—not octal.

Quick Start

1. Open C6461_front_panel_pc_fault_fixed.html in Firefox, Chrome, or Edge. No installation or Internet connection is required.
2. Click Init. The simulator first clears memory, registers, condition codes, faults, the halt state, and the keyboard input queue.
3. Select an assembler-generated load file containing an octal address and octal word on each nonblank line.
4. Wait for the “Memory load complete” message.
5. Set PC to the program entry address. You may type the octal value directly or set the front-panel switches and click the PC LD button.
6. Click SS to execute one instruction or Run to execute continuously.
7. When execution stops, inspect PC, MAR, MBR, IR, the registers, CC/MFR, memory locations, and terminal output.

1. Starting the Simulator

The simulator is a single HTML file. It runs locally inside the browser and does not send programs or data to a server.

Opening the application

- Place the simulator HTML file and the test/load files in a convenient course folder.
- Double-click the HTML file, or right-click it and choose a browser.
- Maximize the browser window or use horizontal scrolling if the full front panel and terminal do not fit on the screen.

Main screen areas

Area	Purpose
General and index registers	Displays and manually loads GPR 0-3 and IXR 1-3.
Processor registers	Displays and loads PC, MAR, and MBR; displays the current IR, MFR, and CC.

Execution controls	Store, St+, Load, Ld+, Run/Stop, SS, manual HLT, and Init.
Switch register	Builds one 16-bit word by clicking bits 0-15; shows the binary and octal value.
Console Terminal	Queues keyboard input and displays printer output.

2. Registers and Displays

Display	Format	Purpose
GPR 0-3	6-digit octal	Four 16-bit general-purpose registers used by arithmetic, logic, memory, branch, and I/O instructions.
IXR 1-3	6-digit octal	Three index registers used in effective-address calculation. There is no physical IXR 0.
PC	4-digit octal	Program Counter. Holds the address of the next instruction to fetch.
MAR	4-digit octal	Memory Address Register. Selects the memory location used by the front-panel Store/Load controls and shows the most recent instruction-fetch address.
MBR	6-digit octal	Memory Buffer Register. Holds the front-panel word to store or the word just loaded/fetched.
IR	6-digit octal	Instruction Register. Shows the instruction currently being executed.
MFR	4-bit binary	Machine Fault Register. Shows the active machine-fault bit pattern.
CC	4-bit binary	Condition Code register. Shows overflow, underflow, divide-by-zero, and equality flags.

Condition-code display

Displayed bit	Meaning	Typical setter
CC(0), leftmost	Overflow	Add/subtract overflow and left-shift bits lost.
CC(1)	Underflow	Add/subtract underflow and right-shift bits lost.
CC(2)	Divide by zero	DVD when the divisor is zero.
CC(3), rightmost	Equal or not equal	TRR sets the bit when the compared registers are equal.

3. Using the Switch Register and LD Buttons

The bottom of the panel is a 16-bit switch register. Bits are numbered 0 through 15 from left to right and are grouped according to the instruction format: Operation, GPR, IXR, I, and Address.

1. Click a numbered bit to toggle it between 0 and 1.
2. Read the complete binary word and its six-digit octal equivalent below the switches.
3. Click the LD button beside GPR, IXR, PC, MAR, or MBR to copy the switch word into that register. PC and MAR receive only the low 12 bits.
4. Register boxes are also editable. When entering a value directly, use octal digits 0 through 7.

The LD buttons load a register from the switch bank. They do not read from memory. Use the separate Load or Ld+ controls to read memory into MBR.

4. Front-Panel Memory Controls

Control	Action
Store	Writes the current MBR word into memory at the address in MAR. MAR is unchanged.
St+	Stores MBR into memory at MAR, then increments MAR by one.
Load	Reads memory at MAR into MBR. MAR is unchanged.
Ld+	Increments MAR by one first, then reads that new memory location into MBR.

Examining one memory location

1. Enter the desired octal address in MAR, or set the switches and click MAR LD.
2. Click Load.
3. Read the six-digit octal contents in MBR.

Changing one memory location

1. Set MAR to the desired octal address.
2. Set MBR to the desired six-digit octal word.
3. Click Store.
4. Click Load again if you want to verify the stored value.

5. Loading a Program with Init

Init performs a complete machine reset and then opens a file chooser. Because the reset happens before the chooser opens, canceling the chooser still leaves the simulator reset.

Required load-file format

Each generated line must contain an octal memory address followed by an octal 16-bit value. Blank lines are ignored. A semicolon begins a comment and the rest of that line is ignored.

```
000006 000012
```

```
000007 000003
000010 002000   ; optional comment
```

Requirement	Rule
Address	Octal, within memory locations 0000 through 3777 (decimal 0-2047).
Data word	Octal, within 000000 through 177777 (16 bits).
Columns	At least two whitespace-separated fields: address and value.
Source file	Do not load the human-readable assembly source. Load the assembler-generated two-column file.

File chooser note: The current chooser advertises .asm and .txt files. If a .load file is hidden, choose All Files or make a copy with a .txt extension. The file contents remain plain text.

6. Executing a Program

Control	Behavior
SS	Fetches and executes exactly one instruction, unless the machine is already halted.
Run	Starts continuous execution. While running, the button changes to Stop.
Stop	Stops continuous execution after the current timer interval; it does not set the machine halt state.
HLT button	Manually stops execution and lights red. A program HLT instruction lights the same indicator.
Init	Stops execution and resets the complete simulator before loading a new file.

Instruction fetch and the PC

During a normal instruction step, the simulator copies PC to MAR, reads memory into MBR and IR, increments PC, and then executes the instruction. A branch or jump may replace the incremented PC with a new address.

After a HLT instruction, PC normally points to the address following HLT because the fetch increment occurred before the instruction halted the machine.

Restarting after a halt

Clicking Run while halted clears the halt indication and begins execution from the current PC. SS by itself does not execute while the machine remains halted. Init is the cleanest way to start a new test.

7. Console Input and Output

Device ID	Meaning	Instructions
0	Keyboard input	CHK and IN
1	Printer/terminal output	CHK and OUT
2	Card-reader style input queue	CHK and IN

Sending input

1. Type text in Line Input.
2. Press Enter or click Send Line.
3. The terminal echoes the line with a leading > symbol.
4. The simulator places every character into the input queue and then adds one newline character.

The newline character is decimal 10, octal 000012. IN removes exactly one queued character. A program that reads a complete line should continue until it consumes the newline.

CHK, IN, and OUT behavior

Instruction	Current simulator behavior
CHK r,0 or CHK r,2	Places 1 in the selected GPR when at least one input character is waiting; otherwise places 0.
CHK r,1	Places 1 in the selected GPR because the printer is always ready.
IN r,0 or IN r,2	Removes one character from the queue and places its low 8-bit code in the selected GPR. If the queue is empty, it places 0.
OUT r,1	Writes the low 8 bits of the selected GPR as one terminal character.

Typical polling logic

```
Wait:  CHK 0,0        ; R0 = 1 when keyboard input is waiting
       JZ  0,0,Wait   ; poll again while R0 is zero
       IN  1,0        ; read one character into R1
       ; compare R1 with decimal 10 / octal 000012 for end of line
```

The terminal Clear button clears only the visible Output/Echo window. It does not remove unread characters from the input queue. Init clears the queue.

8. Addressing and Protected Memory

The simulator contains 2,048 memory words, addressed from decimal 0 through 2047 (octal 0000 through 3777). Normal memory-reference instructions begin with the five-bit address field, optionally add an index register, and optionally perform one level of indirect addressing.

Case	Behavior
No indexing, direct	EA is the instruction address field.
Indexed, direct	EA is the address field plus IXR 1, 2, or 3.
Indirect	The word at the first EA supplies the final 12-bit address.
LDX/STX	The IX field selects the index register and is not also added to the address.
LDA	Loads the computed effective address itself instead of reading memory at that address.

Reserved locations

Location	Use in the current simulator
0	Contains the base address of the TRAP vector table.
1	Contains the entry address of the machine-fault handler.
2	Receives the return PC for a TRAP.
3	Reserved by the architecture.
4	Receives the address of the instruction that caused a machine fault.
5	Reserved by the architecture.

Normal data accesses to locations 0-5 cause a reserved-location fault. Instruction fetches from those locations are permitted, and LDA may produce a reserved address because it does not read or write that location.

9. TRAP and Machine-Fault Handling

TRAP

1. The simulator stores the already-incremented return PC in memory location 2.
2. Memory location 0 supplies the base address of the TRAP table.
3. The four-bit TRAP code selects one of as many as 16 table entries.
4. The selected table entry supplies the address of the TRAP routine, which becomes the new PC.

Machine faults

When a machine fault occurs, the simulator sets MFR, stores the address of the faulting instruction in memory location 4, and loads PC from the handler address stored in memory location 1. If location 1 does not contain a valid handler address of 0006 or greater, the simulator halts.

MFR display	Fault
0001	Illegal data access to a reserved location.
0010	Illegal TRAP code.
0100	Illegal or unsupported operation. Invalid register/device combinations are also treated as illegal operations.
1000	Address beyond installed memory.

Saved-PC correction in this version: if an illegal instruction is located at octal 0700, memory location 4 receives 000700—not the already-incremented 000701.

10. Implemented Instruction Groups

Group	Implemented mnemonics
Miscellaneous	HLT, TRAP
Load/store	LDR, STR, LDA, LDX, STX
Arithmetic	AMR, SMR, AIR, SIR, MLT, DVD
Transfer of control	JZ, JNE, JCC, JMA, JSR, RFS, SOB, JGE
Logical/compare	TRR, AND, ORR, NOT
Shift/rotate	SRC, RRC
Input/output	IN, OUT, CHK

Not implemented in the current simulator: FADD, FSUB, VADD, VSUB, CNVRT, LDFR, and STFR. Executing one of these opcodes produces MFR 0100 and transfers control to the machine-fault handler.

11. Using the Comprehensive Test Suite

Use CSCI6461_all_instruction_test.load with Init. The matching source and listing are CSCI6461_all_instruction_test.asm and CSCI6461_all_instruction_test.lst.

Test	Starting PC	Expected result

Core self-test	000006	Memory 000425 becomes 000001, followed by HLT.
Manual I/O test	000316	CHK, IN, and OUT operate on one queued character; final PC is 000322 after HLT.
Manual TRAP test	000322	TRAP 0 reaches the routine at 000640; its HLT leaves PC at 000641.
FP/vector test	000341	The suite expects memory 000520 = 000001 only after Phase IV instructions are implemented. The current simulator instead faults on the first unsupported FP/vector opcode.

Manual I/O test procedure

1. Click Init and load CSCI6461_all_instruction_test.load.
2. Type A in Line Input and press Enter. The queue now contains A followed by newline.
3. Set PC to 0316.
4. Click SS once: PC should be 0317 and GPR 0 should be 000001.
5. Click SS again: PC should be 0320 and GPR 1 should be 000101, the octal character code for A.
6. Click SS again: A should appear in terminal output and PC should be 0321.
7. Click SS once more: HLT should light red and PC should be 0322.

Illegal-opcode fault test

1. Load the test file so memory location 1 contains the fault-handler address.
2. Store octal 176000 in memory location 0700.
3. Set PC to 0700 and click Run.
4. The machine should transfer to the handler at 0660, set MFR to 0100, and halt after the handler instruction.
5. Load memory location 0004. MBR should show 000700, the address of the faulting instruction.

12. Common Problems

Symptom	Likely cause and correction
The .load file is not visible	Choose All Files or copy/rename the plain-text file with a .txt extension.
Load failed: format error	Use the assembler-generated two-column octal file, not assembly source or a listing file.
SS does nothing	The simulator is halted. Use Init for a clean restart, or Run to clear halt and continue from the current PC.
Program stops at fault handler 0660	Check MFR. An unsupported FP/vector opcode or another illegal operation produces 0100.
IN reads 000000	The input queue was empty. Send a line before executing IN, or poll with CHK.

A later input operation immediately sees data	A prior program may have left the newline in the queue. Consume it in the program or reload with Init.
Clear did not remove pending input	Clear affects only visible terminal output. Init clears the input queue.
Unexpected final PC after HLT	The fetch cycle increments PC before HLT executes, so PC points to the next location.
Memory 0004 contains the wrong address	Use the pc-fault-fixed simulator version. It saves the faulting instruction address.

13. Recommended Test Workflow

1. Assemble the source and correct every assembler error.
2. Save the .asm, .lst, and .load files together.
3. Open the current simulator and click Init.
4. Load the two-column octal file.
5. Set the documented entry PC.
6. Single-step the first few instructions and compare PC, IR, registers, CC, and memory with expected values.
7. Run continuously only after the early steps are correct.
8. Record the final PC, MFR/CC, registers, changed memory locations, and terminal output.

Quick Checklist

☐ The current simulator file is C6461_front_panel_pc_fault_fixed.html.

☐ The load file contains two octal columns and no assembly syntax.

☐ PC and MAR entries are octal addresses.

☐ GPR, IXR, MBR, and IR values are octal words.

☐ MFR and CC are read as four binary bits.

☐ The intended program entry address is loaded into PC.

☐ Keyboard input is queued before a non-polling IN test is executed.

☐ Line-oriented input consumes the trailing newline.

☐ Memory location 4 is checked after a machine fault when the saved fault address matters.

☐ FP/vector instructions are not expected to run in the current version.