

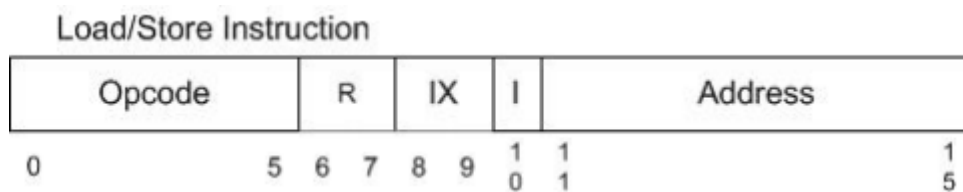
DESIGN DOCUMENT

Sagar Sheth
Ye Wang
Yogesh Mahato
Mayur Chaturvedi

The project's goal is to create an assembly language-based simulator of a modest traditional CISC computer. The project's first phase will have the following features:

- 4 General Purpose Registers (GPRs) – each 16 bits in length [**GPR0, GPR1, GPR2, GPR3**]
- 3 Index Registers – 16 bits in length [**IXR1, IXR2, IXR3**]
- 16-bit words
- Memory of 2048 words, expandable to 4096 words
- Word addressable

Load/Store Instruction:



Opcode: 6 bits – Specifies one of 64 possible instructions; Phase 1 will have 5

Load/Store Opcodes. R : 2 bits – GPR0(00), GPR1(01), GPR2(10), GPR3(11) General Purpose Registers.

IX : 2 bits – IXR1(01), IXR2(10), IXR3(11) Indexed Registers.

I : 1 bit – I=0 specifies indirect addressing, or otherwise no indirect addressing. Address: 5 bits – Specifies one of 32 locations.

Effective address to execute various Load/Store instructions will be computed as follows: Effective Address (EA) =

I = 0:

IX = 00: content (address field)

IX = 01 or 10 or 11: content(IX) + content of address field

I = 1:

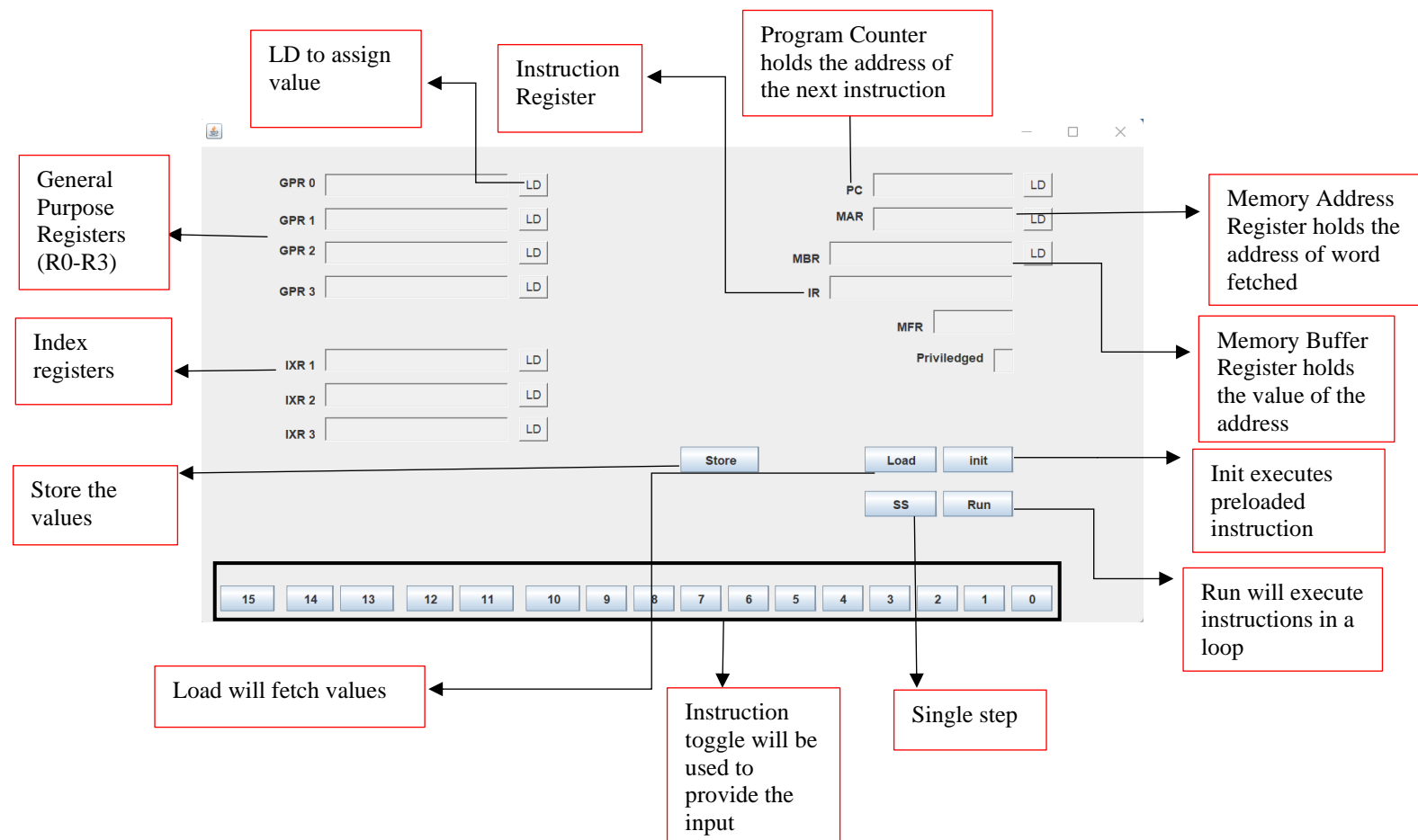
IX = 00: content (content (address field))

IX = 01 or 10 or 11: content(content(IX) + content of address field)

Following are the Load/Store instructions implemented in the Simulator (I is optional):

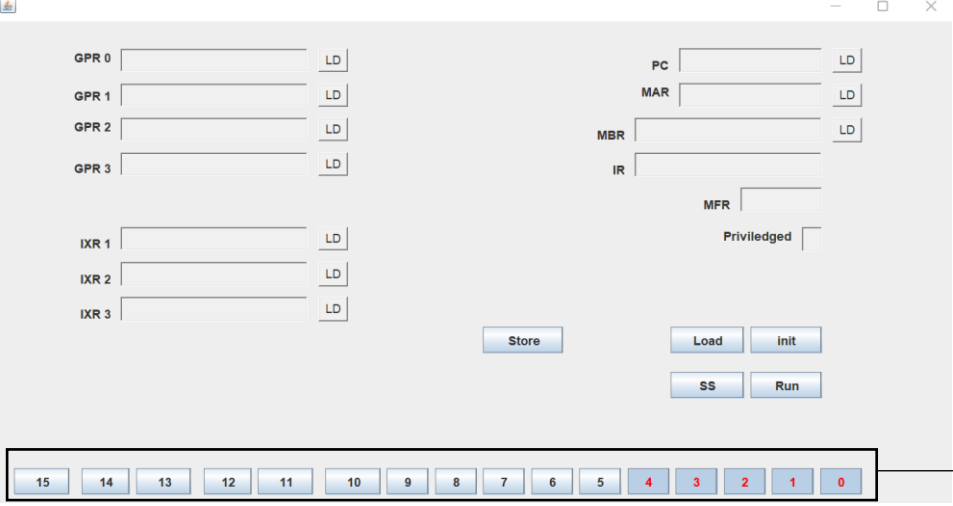
OpCode	Instruction	Description
01	LDR r x I address	Load Register From Memory, r = 0..3 $r \leftarrow c(EA)$
02	STR r x I address	Store Register To Memory, r = 0..3 $Memory(EA) \leftarrow c(r)$
03	LDA r x I address	Load Register with Address, r = 0..3 $r \leftarrow EA$
41	LDX x I address	Load Index Register from Memory, x = 1..3 $Xx \leftarrow c(EA)$
42	STX x I address	Store Index Register to Memory. X = 1..3 $Memory(EA) \leftarrow c(Xx)$

Simulator Design:



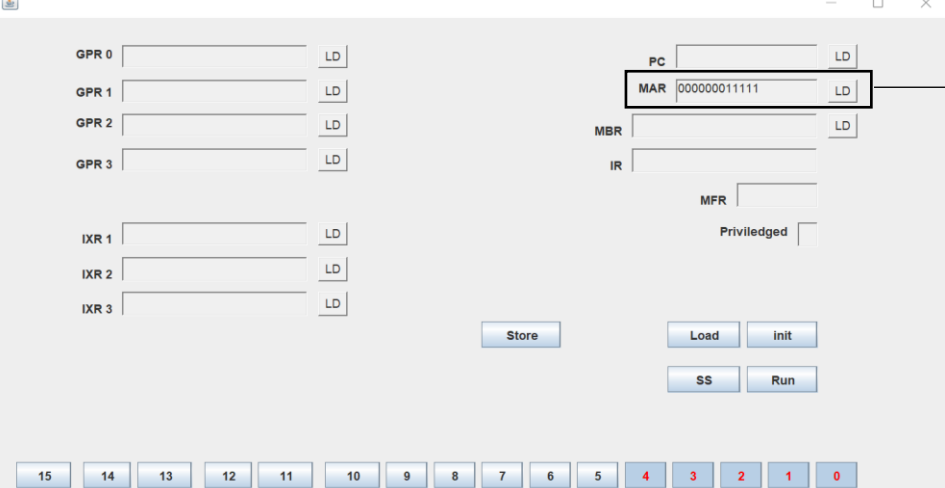
Steps to add instruction to memory:

1. Input MAR instruction in Instruction Toggle, as highlighted below (fields highlighted in red is 1, rest fields are considered 0)



Populate fields in Instruction toggle

2. Click the “LD” button next to MAR to Load it to the MAR register.



LD Button next to MAR to assign value to MAR

- Do the above steps for MBR also:

Enter fields in Instruction toggle, click on LD next to MBR

- Click the “Store” button to store the MBR instruction entered to the MAR memory location

Steps to load instruction from the memory:

- Enter fields under instruction toggle, click on “LD” next to MAR, this will populate fields in MAR
- Now, click on “Load” button, this will populate MBR field for the corresponding MAR entered earlier

Enter stored values for MAR, click on LD

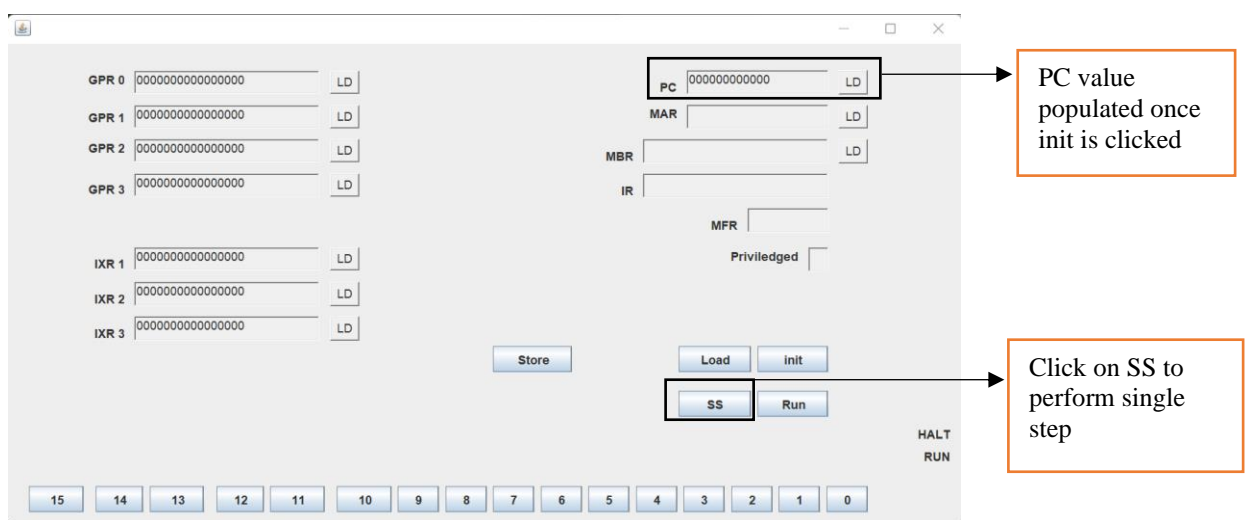
Post LD, click on Load to populate MBR for the mentioned MAR

Steps to load IPL/ init file:

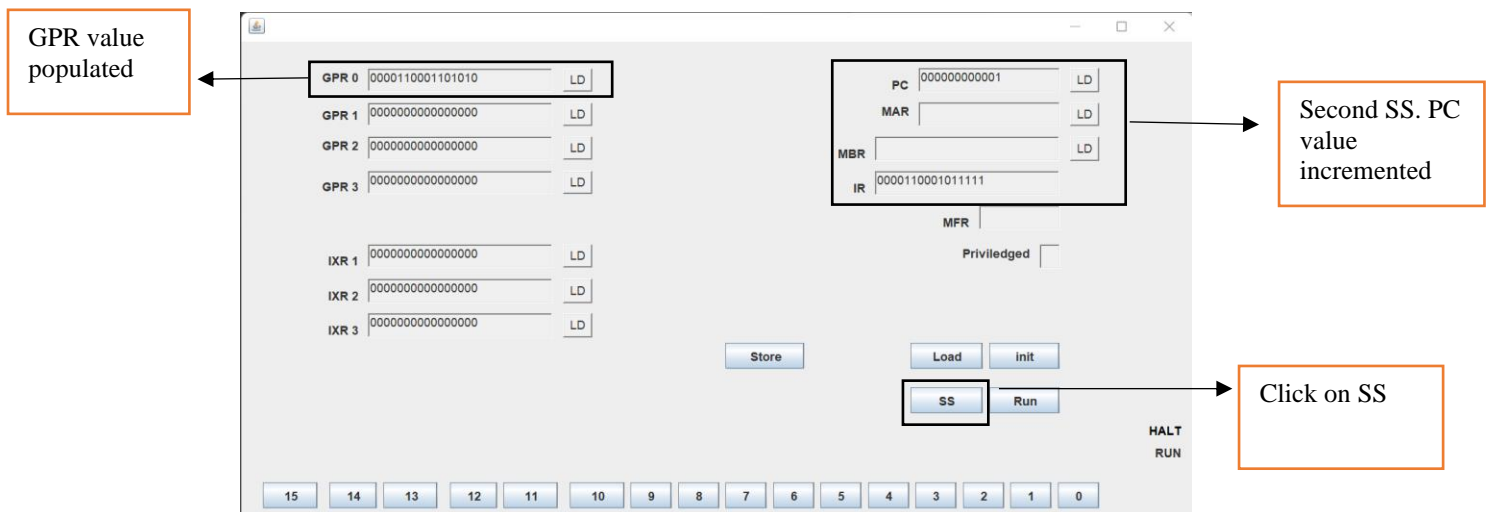
7. Click on init button, this will fetch the default instruction file from the path entered in the init process.
8. Once the IPL/ init file is loaded, the PC will be loaded with first instruction's memory address

How to Single Step:

9. Init button will automatically load PC value stored, in case of custom PC values to be executed, enter the fields in instruction toggle and click on "LD" button next to PC

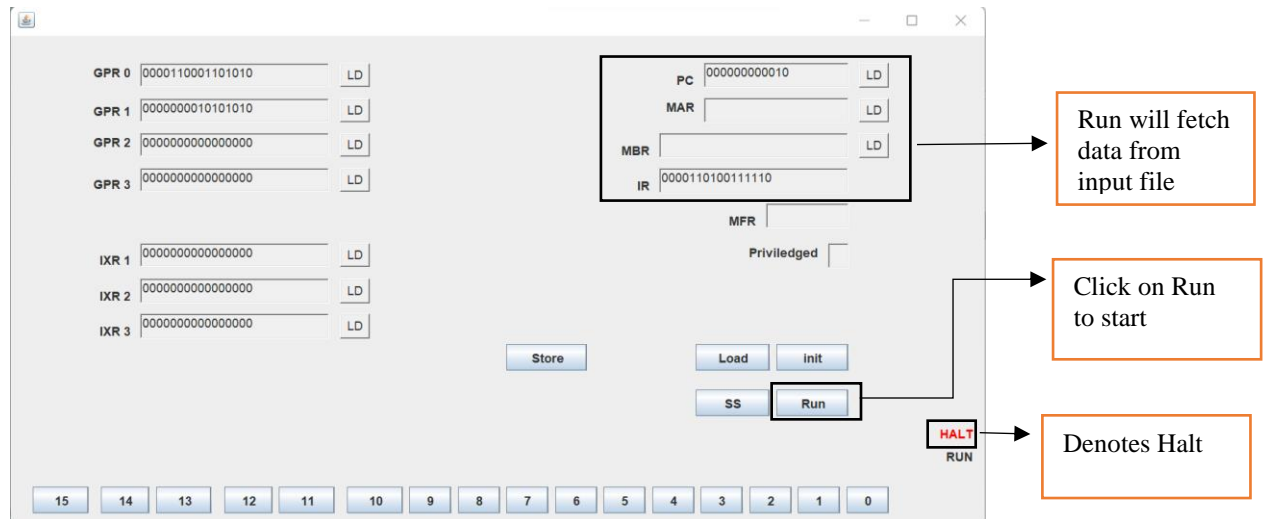


10. Click on the "SS" button to Single step, there will be a delay of 1 sec for registers to be loaded with the values. The PC will increment by 1.



Steps to execute “Run” process:

11. Click on Init button.
12. To perform Single Step (SS) multiple times or until it encounters HALT, “Run” can be used to execute all the instructions from the input text file.
13. HALT button will turn red when it encounters Halt event.

**Phase II****Updated UI elements:**

1. Cache:
2. Printer: Used to Print
3. Console: Displays output when instructions are executed
4. Keyboard: To get input from the user

Execution:

1. **LDR r, x, address[,I] Load Register From Memory, $r = 0..3$ $r \leftarrow c(EA)$ note that EA is computed as given above**
Opcode: 01
 - i. Load MAR with B: 0111 and with MBR value B: 1100, store.
 - ii. Load MBR Value as B: 0000010000000111 with Opcode being 1
 - iii. Load MAR with B: 0100
 - iv. Load PC with B: 0100
 - v. Click on store
 - vi. Click on SS, then PC will be incremented to B:0101 and GPR0 is populated with value with B: 1100, which is the value at the effective address B: 0111.

The screenshot shows a simulator window with the following components:

- Registers (Left):**
 - GPR 0: 0000000000001100
 - GPR 1: 0000000000000000
 - GPR 2: 0000000000000000
 - GPR 3: 0000000000000000
 - IXR 1: 0000000000000000
 - IXR 2: 0000000000000000
 - IXR 3: 0000000000000000
- Control Elements (Right):**
 - PC: 000000001001
 - MAR: 000000001000
 - MBR: 0000100000001111
 - IR: 0000010000000111
 - MFR: (empty)
 - Privileged: ☐
- Buttons:** Store, Load, Init, SS, Run.
- Bottom Bar:** A row of buttons numbered 15 down to 0. Button 2 is highlighted in red.
- Status:** HALT and RUN indicators on the right.

2. STR r, x, address[,I] Store Register To Memory, $r = 0..3$ Memory(EA) $\leftarrow c(r)$

Opcode: 02

- i. Load the GPR0 with B: 0000000000000011
- ii. Load MBR with B: 0000100000001111, with Opcode being 2
- iii. Load MAR with B: 1000
- iv. Set PC to B:1000
- v. Click on store
- vi. Click SS

The screenshot shows the simulator window after the execution of the STR instruction. The state is as follows:

- Registers (Left):**
 - GPR 0: 0000000000000011
 - GPR 1: 0000000000000000
 - GPR 2: 0000000000000000
 - GPR 3: 0000000000000000
 - IXR 1: 0000000000000000
 - IXR 2: 0000000000000000
 - IXR 3: 0000000000000000
- Control Elements (Right):**
 - PC: 000000001001
 - MAR: 000000001000
 - MBR: 0000100000001111
 - IR: 0000100000000111
 - MFR: (empty)
 - Privileged: ☐
- Buttons:** Store, Load, Init, SS, Run.
- Bottom Bar:** A row of buttons numbered 15 down to 0. Button 3 is highlighted in red.
- Status:** HALT and RUN indicators on the right.

2. LDA r, x, address[,I] Load Register with Address, r = 0..3 $r \leftarrow EA$ **Opcode: 03**

- i. Load MBR with B: 0000110000001000
- ii. Load MAR with B: 0000000000110
- iii. Load PC with B: 000000000110
- iv. Click on store
- v. Click SS
- vi. GPR0 is loaded with effective address as B: 1000

The screenshot shows a simulator window with the following components:

- Registers (Left):**
 - GPR 0: 0000000000001000 (LD)
 - GPR 1: 0000000000000000 (LD)
 - GPR 2: 0000000000000000 (LD)
 - GPR 3: 0000000000000000 (LD)
 - IXR 1: 0000000000000000 (LD)
 - IXR 2: 0000000000000000 (LD)
 - IXR 3: 0000000000000000 (LD)
- Control Units (Right):**
 - PC: 000000000111 (LD)
 - MAR: 000000000110 (LD)
 - MBR: 0000110000001000 (LD)
 - IR: 0000110000001000
 - MFR: (empty)
 - Privileged: ☐
- Buttons:** Store, Load, Init, SS, Run.
- Status:** HALT, RUN.
- Address Bus:** A row of buttons labeled 15 down to 0.

5. JZ r, x, address[,I] Jump If Zero: If $c(r) = 0$, then $PC \leftarrow EA$ Else $PC \leftarrow PC+1$ **Opcode: 10**

- i. Load MBR with B: 0010000000001111
- ii. Load MAR with B: 0011
- iii. Load PC with B: 0011 and Set GPR 0 to 0
- iv. Click on store
- v. Click on SS, and PC is Updated as $c(r) = 0$

The simulator interface displays the following components:

- General Purpose Registers (GPR):** GPR 0, GPR 1, GPR 2, GPR 3, each with a value of 0000000000000000 and an LD button.
- Instruction Registers (IXR):** IXR 1, IXR 2, IXR 3, each with a value of 0000000000000000 and an LD button.
- Control Units:** PC (000000001111), MAR (00000000011), MBR (0010000000001111), and IR (0010000000001111), each with an LD button.
- Other:** MFR (empty), Privileged (checkbox), and buttons for Store, Load, Init, SS, and Run.
- Instruction Queue:** A row of 16 buttons labeled 15 down to 0.
- Status:** HALT and RUN indicators.

6. JNE r, x, address[I] Jump If Not Equal: If $c(r) \neq 0$, then $PC \leftarrow EA$ Else $PC \leftarrow PC + 1$

Opcode: 11

- Load GPR0 with B: 0010
- Load MBR with B: 0010010000001000
- Load MAR with B: 1011
- Load PC with B: 1011
- Click on store
- Click SS
- PC is Updated as $c(r) \neq 0$, then $PC \leftarrow EA$

The simulator interface shows the state after executing the JNE instruction:

- General Purpose Registers (GPR):** GPR 0 now contains 0000000000000010. GPR 1, GPR 2, and GPR 3 remain at 0000000000000000.
- Instruction Registers (IXR):** IXR 1 now contains 0000000000001011. IXR 2 and IXR 3 remain at 0000000000000000.
- Control Units:** PC is updated to 000000001100. MAR remains at 000000001011. MBR is updated to 0010010000001000. IR remains at 0010010000001000.
- Other:** MFR (empty), Privileged (checkbox), and buttons for Store, Load, Init, SS, and Run.
- Instruction Queue:** The queue is updated, with button 3 highlighted in red.
- Status:** HALT and RUN indicators.

8. JMA x, address[I] Unconditional Jump To Address PC <- EA, Note: r is ignored in this instruction

Opcode: 13

- i. Load MBR with B: 0010110000001111
- ii. Load MAR with B: 0100
- iii. Set PC to B: 0100
- iv. Click on store
- v. Click SS
- vi. PC is Updated as the effective address value to B: 1111

The screenshot shows a simulator window with the following components:

- Registers:**
 - GPR 0, GPR 1, GPR 2, GPR 3: Each has a value of 0000000000000000 and an LD button.
 - IXR 1, IXR 2, IXR 3: Each has a value of 0000000000000000 and an LD button.
- Control and Status:**
 - PC: 0000000000001111 with an LD button.
 - MAR: 000000000100 with an LD button.
 - MBR: 0010110000001111 with an LD button.
 - IR: 0010110000001111
 - MFR: (empty)
 - Privileged: ☐
- Buttons:** Store, Load, Init, SS, Run.
- Status:** HALT, RUN.
- Address Bus:** A row of buttons labeled 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0.

9. JSR x, address[I] Jump and Save Return Address: R3 <- PC+1; PC <- EA R0 should contain pointer to arguments. Argument list should end with -1 (all 1s) value

Opcode: 14

- i. Load MBR with B: 0011000000001111
- ii. Load MAR with B: 0111
- iii. Set PC to B: 0111
- iv. Click on store
- v. Click SS
- vi. PC is Updated as B:1111 and GPR3 becomes PC+1, i.e., B: 1000

The simulator interface displays the following components:

- General Purpose Registers (GPR):** GPR 0 to GPR 3, each with a value field and an LD button. GPR 3 contains 0000000000001000.
- Instruction Registers (IXR):** IXR 1 to IXR 3, each with a value field and an LD button. All contain 0000000000000000.
- Control Registers:** PC (000000001111), MAR (00000000111), MBR (0011000000001111), IR (0000000000000000), and MFR (empty).
- Privileged:** A checkbox.
- Buttons:** Store, Load, init, SS, and Run.
- Opcode Display:** A row of 16 buttons labeled 15 to 0. Buttons 3, 2, 1, and 0 are highlighted in red.
- Status:** HALT and RUN indicators.

10. RFS Immed Return From Subroutine w/ return code as Immed portion (optional) stored in the instruction's address field. $R0 \leftarrow \text{Immed}$; $PC \leftarrow c(R3)$ IX, I fields are ignored.

Opcode: 15

- Load GPR3 with B: 0000
- Load MBR with B: 0011010000001111
- Load MAR with B: 0111
- Load PC with B: 0111
- Click on store
- Click SS
- GPR0 becomes B:1111, which is EA, PC becomes B:0000 which is content of GP3.

The simulator interface shows the state after execution:

- General Purpose Registers (GPR):** GPR 0 now contains 0000000000001111. GPR 1 to GPR 3 remain 0000000000000000.
- Instruction Registers (IXR):** IXR 1 to IXR 3 remain 0000000000000000.
- Control Registers:** PC is 0000000000000000, MAR is 00000000111, MBR is 0011010000001111, and IR is 0011010000001111.
- Privileged:** The checkbox remains unchecked.
- Buttons:** Store, Load, init, SS, and Run.
- Opcode Display:** The same row of 16 buttons, with 3, 2, 1, and 0 highlighted in red.
- Status:** HALT and RUN indicators.

11. SOB r, x, address[,I] Subtract One and Branch. R = 0..3 $r \leftarrow c(r) - 1$ If $c(r) > 0$, PC \leftarrow EA; Else PC \leftarrow PC + 1

Opcode: 16

- i. Load GPR0 with B: 0011
- ii. Load MBR with B: 0011100000001111
- iii. Load MAR with B: 0111
- iv. Load PC with B: 0111
- v. Click on store
- vi. Click SS
- vii. PC is updated to B: 1111, GPR0 is subtracted by 1 and becomes B: 0010

The screenshot shows a simulator window with the following components:

- Registers (Left):**
 - GPR 0: 0000000000000010 (LD)
 - GPR 1: 0000000000000000 (LD)
 - GPR 2: 0000000000000000 (LD)
 - GPR 3: 0000000000000000 (LD)
 - IXR 1: 0000000000000000 (LD)
 - IXR 2: 0000000000000000 (LD)
 - IXR 3: 0000000000000000 (LD)
- Control and Memory (Right):**
 - PC: 0000000000001111 (LD)
 - MAR: 000000000111 (LD)
 - MBR: 0011100000001111 (LD)
 - IR: 0011100000001111
 - MFR: (empty)
 - Privileged: ☐
- Buttons:** Store, Load, init, SS, Run.
- Status:** HALT, RUN.
- Address Bus (Bottom):** A row of buttons from 15 to 0. Buttons 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, and 0 are blue. Buttons 2 and 1 are red.

12. JGE r,x, address[,I] Jump Greater Than or Equal To: If $c(r) \geq 0$, then PC \leftarrow EA Else PC \leftarrow PC + 1

Opcode: 17

- i. Load GPR0 with B: 0111
- ii. Load MBR with B: 0011110000001111
- iii. Load MAR with B: 0011
- iv. Set PC to B: 0011
- v. Click store
- vi. Click SS
- vii. PC is updated to B: 1111

The simulator interface displays the following components:

- Registers (GPRs and IXRs):**
 - GPR 0: 000000000000111
 - GPR 1: 000000000000000
 - GPR 2: 000000000000000
 - GPR 3: 000000000000000
 - IXR 1: 000000000000000
 - IXR 2: 000000000000000
 - IXR 3: 000000000000000
- Control Units:**
 - PC: 000000000000111
 - MAR: 00000000011
 - MBR: 0011110000001111
 - IR: 0011110000001111
 - MFR: (empty)
 - Privileged: ☐
- Buttons:** Store, Load, Init, SS, Run.
- Instruction Queue:** A row of 16 buttons labeled 15 down to 0. Button 1 is highlighted in red.
- Status:** HALT and RUN indicators.

13. AMR r, x, address[,I] Add Memory To Register, $r = 0..3$ $r \leftarrow c(r) + c(EA)$

Opcode: 04

- Load GPR1 with B: 0111
- Load MAR with B: 111 and MBR with B: 0110, store
- Load MAR with B: 0010 and MBR with B: 0001000100000111
- Set PC to B: 0010
- Click on store
- Click SS
- GPR1 becomes $c(r) + c(EA) = B: 0110 + B: 0111 = B: 1101$

The simulator interface shows the state after the AMR instruction execution:

- Registers (GPRs and IXRs):**
 - GPR 0: 000000000000000
 - GPR 1: 0000000000001101
 - GPR 2: 000000000000000
 - GPR 3: 000000000000000
 - IXR 1: 000000000000000
 - IXR 2: 000000000000000
 - IXR 3: 000000000000000
- Control Units:**
 - PC: 000000000011
 - MAR: 00000000010
 - MBR: 0001000100000111
 - IR: 0001000100000111
 - MFR: (empty)
 - Privileged: ☐
- Buttons:** Store, Load, Init, SS, Run.
- Instruction Queue:** A row of 16 buttons labeled 15 down to 0. Button 1 is highlighted in red.
- Status:** HALT and RUN indicators.

14. SMR r, x, address[,I] Subtract Memory From Register, $r = 0..3$ $r \leftarrow c(r) - c(EA)$

Opcode: 05

- i. Load GPR3 with B: 1111
- ii. Load MAR with B: 0111 and MBR with B: 0110, store
- iii. Load MAR with B: 0010 and set MBR with B: 0001011100000111
- iv. Set PC to B: 0010
- v. Click on store
- vi. Click SS
- vii. GPR3 becomes $c(r) - c(EA) = B: 1111 - B: 110 = B: 1001$

The screenshot shows a simulator interface with the following components:

- Registers:**
 - GPR 0: 0000000000000000 LD
 - GPR 1: 0000000000000000 LD
 - GPR 2: 0000000000000000 LD
 - GPR 3: 0000000000001001 LD
 - IXR 1: 0000000000000000 LD
 - IXR 2: 0000000000000000 LD
 - IXR 3: 0000000000000000 LD
- Control and Memory:**
 - PC: 000000000011 LD
 - MAR: 000000000001 LD
 - MBR: 0001011100000111 LD
 - IR: 0001011100000111
 - MFR: (empty)
 - Privileged: ☐
- Buttons:** Store, Load, Init, SS, Run.
- Status:** HALT, RUN.
- Address Bus:** A row of buttons from 15 to 0, with button 1 highlighted in red.

15. AIR r, immed Add Immediate to Register, $r = 0..3$ $r \leftarrow c(r) + \text{Immed}$ Note: 1. if Immed = 0, does nothing 2. if $c(r) = 0$, loads r with Immed IX and I are ignored in this instruction

Opcode: 06

- i. Load GPR0 with B: 0111
- ii. Load MAR with B: 0011 and MBR with B: 0001100000001111
- iii. Set PC to B: 0011
- iv. Click on store
- v. Click SS
- vi. GPR0 becomes $c(r) + \text{immed} = B: 0111 + B: 1111 = B: 10110$

The simulator interface displays the following components:

- General Purpose Registers (GPR):** GPR 0 to GPR 3, each with a value field and an LD button. GPR 0 contains 000000000010110.
- Instruction Registers (IXR):** IXR 1 to IXR 3, each with a value field and an LD button. All contain 0000000000000000.
- Control Registers:** PC (000000000100), MAR (00000000011), MBR (000110000001111), and IR (000110000001111), each with a value field and an LD button.
- Buttons:** Store, Load, Init, SS, and Run.
- Status:** Privileged checkbox (unchecked), and HALT/RUN indicators.
- Address Bus:** A row of 16 buttons labeled 15 down to 0.

16. SIR r, immed Subtract Immediate from Register, r = 0..3 $r \leftarrow c(r) - \text{Immed}$ - Immed Note: 1. if Immed = 0, does nothing 2. if $c(r) = 0$, loads r1 with $-(\text{Immed})$ IX and I are ignored in this instruction

Opcode: 07

- Load GPR2 with B: 11111
- Load MAR with B: 0010 and MBR with B: 0001111000000111
- Set PC to B: 0010
- Click on store
- Click SS
- GPR2 becomes $c(r) - \text{immediate value} = B: 11111 - B: 00111 = B: 11000$

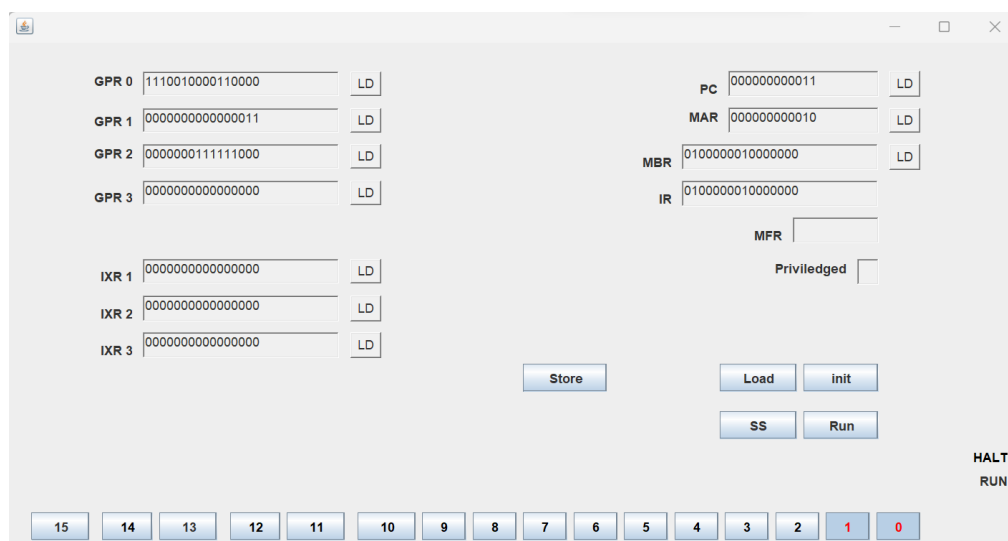
The simulator interface shows the state after the instruction execution:

- General Purpose Registers (GPR):** GPR 0 (0000000000000000), GPR 1 (0000000000000000), GPR 2 (000000000011000), GPR 3 (0000000000000000).
- Instruction Registers (IXR):** IXR 1 to IXR 3 (all 0000000000000000).
- Control Registers:** PC (00000000011), MAR (00000000010), MBR (0001111000000111), and IR (0001111000000111).
- Buttons:** Store, Load, Init, SS, and Run.
- Status:** Privileged checkbox (unchecked), and HALT/RUN indicators.
- Address Bus:** A row of 16 buttons labeled 15 down to 0.

17. MLT rx,ry Multiply Register by Register rx, $rx+1 \leftarrow c(rx) * c(ry)$ rx must be 0 or 2 ry must be 0 or 2

Opcode: 20

- i. Load GPR0 with B: 0000000011111010
- ii. Load GPR2 with B: 0000000011111000
- iii. Load MAR with B: 0010 and MBR to B: 0100000010000000
- iv. Set PC to B: 0010
- v. Click on store
- vi. Click SS
- vii. Result is B: $11111010 * B: 11111000 = B: 111110010000110000$
- viii. First 4 bits are stored in rx which is GPR0, other 4 bits are stored at rx+1 which is GPR1



18. DVD rx,ry Divide Register by Register rx, $rx+1 \leftarrow c(rx) / c(ry)$ rx must be 0 or 2 rx contains the quotient; rx+1 contains the remainder ry must be 0 or 2 If $c(ry) = 0$, set cc(2) to 1 (set DIVZERO flag)

Opcode: 21

- i. Load GPR0 with B: 1011
- ii. Load GPR2 with B: 0011
- iii. Load MAR with B: 0111 and MBR with B: 0100010010000000
- iv. Set PC to B: 0111
- v. Click on store
- vi. Click SS
- vii. Result for B: $1011 / B: 0011$ is Quotient: B: 0011 Remainder: B: 0010
- viii. rx that is GPR0 becomes quotient: B: 0011 and rx+1 that is GPR1 becomes remainder: B: 0010

The simulator interface displays the following components:

- General Purpose Registers (GPR):** GPR 0 to GPR 3, each with a value field and an LD button.
- Instruction Registers (IXR):** IXR 1 to IXR 3, each with a value field and an LD button.
- Control Units:** PC, MAR, MBR, IR, MFR, and Privileged flag, each with a value field and an LD button.
- Buttons:** Store, Load, Init, SS, and Run.
- Status:** HALT and RUN indicators.
- Address Bus:** A row of buttons labeled 15 down to 0.

19. TRR rx, ry Test the Equality of Register and Register If $c(rx) = c(ry)$, set $cc(4) \leftarrow 1$; else, $cc(4) \leftarrow 0$

Opcode: 22

- Load GPR0 with B: 0111
- Load GPR1 with B: 0111
- Load MAR with B: 0010 and MBR with B: 0100100001000000
- Set PC to B: 0010
- Click store
- Click SS

The simulator interface shows the state after executing the instructions:

- General Purpose Registers (GPR):** GPR 0 and GPR 1 now contain the value 0000000000000111. GPR 2 and GPR 3 remain at 0000000000000000.
- Instruction Registers (IXR):** IXR 1, IXR 2, and IXR 3 remain at 0000000000000000.
- Control Units:** PC is 0000000000000111, MAR is 0000000000000111, MBR is 0100100001000000, IR is 0100100001000000, MFR is empty, and Privileged is false.
- Buttons:** Store, Load, Init, SS, and Run.
- Status:** HALT and RUN indicators.
- Address Bus:** A row of buttons labeled 15 down to 0, with button 1 highlighted in red.

20. AND rx, ry Logical And of Register and Register $c(rx) \leftarrow c(rx) \text{ AND } c(ry)$ **Opcode: 23**

- i. Set GPR0 to B: 1101
- ii. Set GPR1 to B: 0100
- iii. Set MAR to B: 0011 and MBR with B: 0100110001000000
- iv. Set PC to B: 0011
- v. Click on store
- vi. Click SS
- vii. GPR0 populates with B: 0100

The screenshot shows a simulator window with the following components:

- Registers (Left):**
 - GPR 0: 0000000000000100 (LD)
 - GPR 1: 0000000000000100 (LD)
 - GPR 2: 0000000000000000 (LD)
 - GPR 3: 0000000000000000 (LD)
 - IXR 1: 0000000000000000 (LD)
 - IXR 2: 0000000000000000 (LD)
 - IXR 3: 0000000000000000 (LD)
- Control and Memory (Right):**
 - PC: 00000000000100 (LD)
 - MAR: 0000000000011 (LD)
 - MBR: 0100110001000000 (LD)
 - IR: 0100110001000000
 - MFR: (empty)
 - Privileged: ☐
- Buttons:** Store, Load, Init, SS, Run.
- Bottom Bar:** A row of 16 buttons labeled 15 down to 0. Buttons 14, 11, 10, and 6 are highlighted in red.
- Status:** HALT and RUN indicators on the right.

21. ORR rx, ry Logical Or of Register and Register $c(rx) \leftarrow c(rx) \text{ OR } c(ry)$ **Opcode: 24**

- i. Load GPR0 with B: 1000
- ii. Load GPR1 with B: 0100
- iii. Load MAR with B: 0111 and MBR with B: 0101000001000000
- iv. Set PC to B: 0111
- v. Click on store
- vi. Click on SS
- vii. GPR0 becomes B: 1100

The simulator interface displays various registers and control units. On the left, General Purpose Registers (GPR 0-3) and Instruction Registers (IXR 1-3) are shown with their current values and a 'LD' button. On the right, the Program Counter (PC), Memory Address Register (MAR), Memory Buffer Register (MBR), Instruction Register (IR), and Memory Function Register (MFR) are displayed. A 'Privileged' checkbox is also present. At the bottom, a row of buttons numbered 15 to 0 is visible, with button 6 highlighted in red. Control buttons include 'Store', 'Load', 'init', 'SS', and 'Run'. The status 'HALT' and 'RUN' are shown in the bottom right corner.

GPR 0	0000000000001100	LD
GPR 1	0000000000000100	LD
GPR 2	0000000000000000	LD
GPR 3	0000000000000000	LD
IXR 1	0000000000000000	LD
IXR 2	0000000000000000	LD
IXR 3	0000000000000000	LD
PC	000000001000	LD
MAR	000000000111	LD
MBR	0101000001000000	LD
IR	0101000001000000	
MFR		
Privileged	<input type="checkbox"/>	

Buttons: Store, Load, init, SS, Run

Status: HALT, RUN

Register Bank: 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

22. NOT rx Logical Not of Register To Register C(rx) ← NOT c(rx)

Opcode: 25

- Load GPR0 with B: 1010101010101010
- Load MAR with B: 0010 and MBR with B: 0101010000000000
- Set PC to B: 0010
- Click on store
- Click SS
- GPR0 populates to B: 1010101010101010

The simulator interface shows the state after executing the NOT instruction. GPR 0 now contains the value 111111111111111010101010. The PC has been updated to 000000000011. The MAR and MBR contain the values specified in the instructions. The status 'HALT' and 'RUN' are shown in the bottom right corner.

GPR 0	111111111111111010101010	LD
GPR 1	0000000000000000	LD
GPR 2	0000000000000000	LD
GPR 3	0000000000000000	LD
IXR 1	0000000000000000	LD
IXR 2	0000000000000000	LD
IXR 3	0000000000000000	LD
PC	000000000011	LD
MAR	000000000010	LD
MBR	0101010000000000	LD
IR	0101010000000000	
MFR		
Privileged	<input type="checkbox"/>	

Buttons: Store, Load, init, SS, Run

Status: HALT, RUN

Register Bank: 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

23. SRC r, count, L/R, A/L Shift Register by Count c(r) is shifted left (L/R =1) or right (L/R = 0) either logically (A/L = 1) or arithmetically (A/L = 0) XX, XXX are ignored Count = 0...15 If Count = 0, no shift occurs

Opcode: 31

- i. Load GPR1 with B: 0100
- ii. Load MAR with B: 0111 to MBR with B: 0110010100000001
- iii. Set PC to B: 0111
- iv. Click on store and SS
- v. Since L/R and A/L = 0 it will be right Shifted arithmetically
- vi. Result is B: 0010 which is stored in GPR1

The screenshot shows a simulator window with the following components:

- Registers (Left):**
 - GPR 0: 0000000000000000 LD
 - GPR 1: 0000000000000010 LD
 - GPR 2: 0000000000000000 LD
 - GPR 3: 0000000000000000 LD
 - IXR 1: 0000000000000000 LD
 - IXR 2: 0000000000000000 LD
 - IXR 3: 0000000000000000 LD
- Control and Memory (Right):**
 - PC: 000000001000 LD
 - MAR: 000000000111 LD
 - MBR: 0110010100000001 LD
 - IR: 0110010100000001
 - MFR: (empty)
 - Privileged: ☐
- Buttons:** Store, Load, init, SS, Run.
- Status:** HALT, RUN.
- Counter (Bottom):** A row of buttons from 15 to 0. Buttons 14, 13, 10, 8, and 0 are highlighted in red.

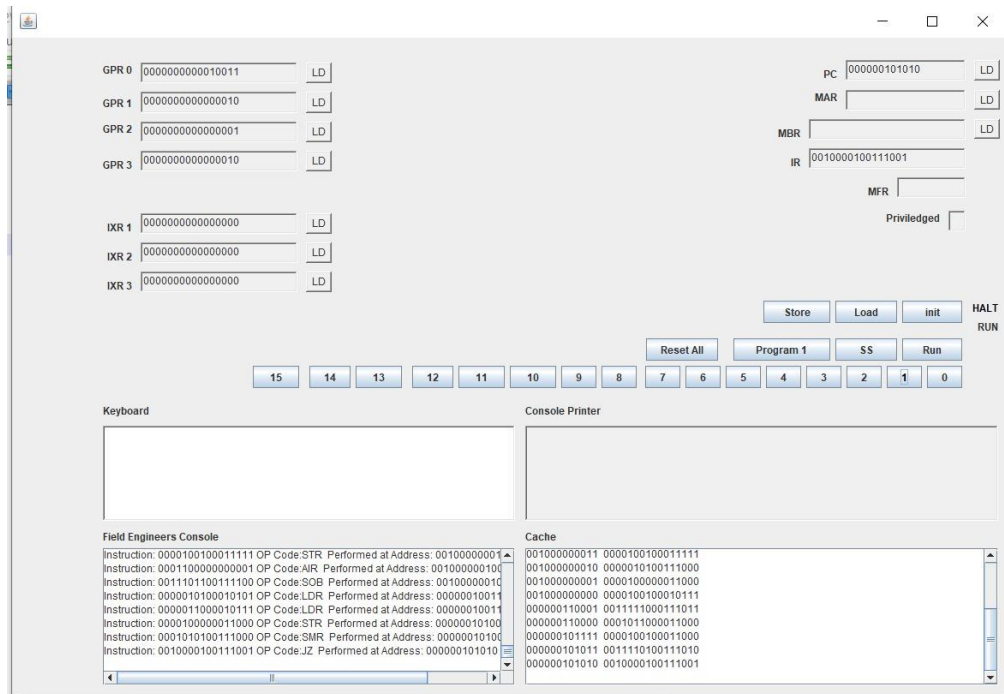
24. RRC r, count, L/R, A/L Rotate Register by Count c(r) is rotated left (L/R = 1) or right (L/R =0) either logically (A/L =1) XX, XXX is ignored Count = 0...15 If Count = 0, no rotate occurs

Opcode: 32

- i. Load GPR1 with B: 0100
- ii. Load MAR with B: 0111 and MBR with B: 0110010100000001
- iii. Set PC to B: 0111
- iv. Click on store
- v. Click SS
- vi. GPR1 is rotated by 1 and becomes B: 0010

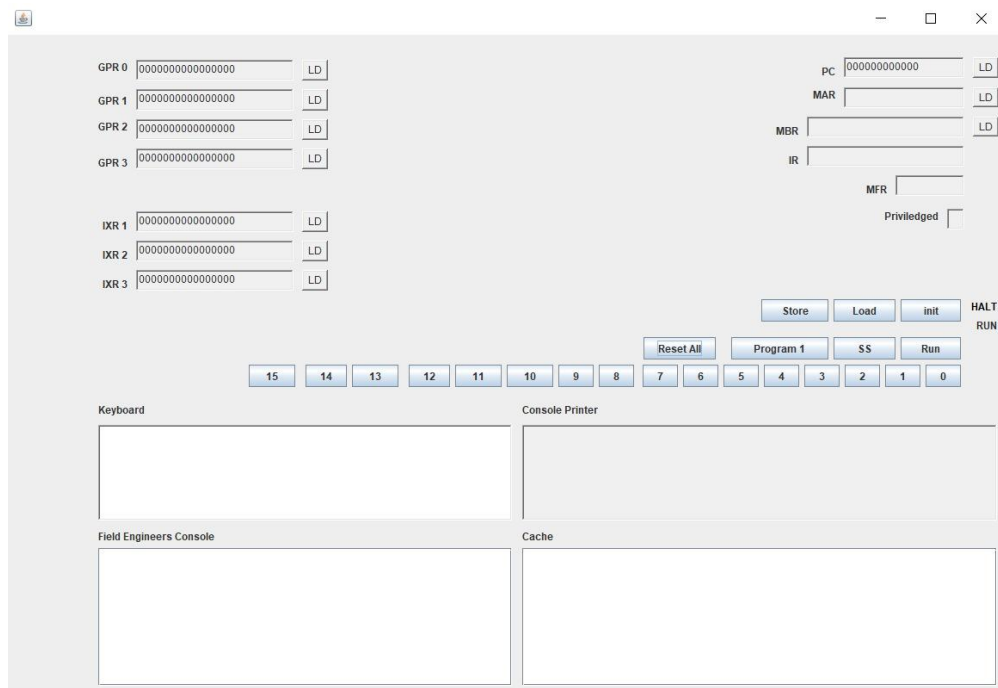
Caching:

Caching operation will display the recently accessed address and memory in a FIFO manner, it will show up to 16 address - memory pairs.



Program 1 Implementation:

1. Click on 'Reset all' button



2. Insert numbers in 'Keyboard' section, delimited by “,”

The screenshot shows a simulator interface with the following components:

- Registers:**
 - GPR 0 to GPR 3: Each has a text input field with '0000000000000000' and an 'LD' button.
 - IXR 1 to IXR 3: Each has a text input field with '0000000000000000' and an 'LD' button.
 - PC: Text input field with '000000100000' and an 'LD' button.
 - MAR: Text input field with an 'LD' button.
 - MBR: Text input field with an 'LD' button.
 - IR: Text input field with an 'LD' button.
 - MFR: Text input field with an 'LD' button.
 - Privileged: A checkbox.
- Buttons:** 'Store', 'Load', 'init', 'Reset All', 'Program 1', 'SS', 'Run', 'HALT', 'RUN'.
- Keyboard:** A row of buttons labeled 15 through 0.
- Console Printer:** A text area showing '10,13,16,19,22,25,28,31,34,37,40,43,46,49,52,55,58,61,64,67,69'.
- Field Engineers Console:** An empty text area.
- Cache:** An empty text area.

3. Load the program by clicking on program1 button
4. Click on run button to execute the program

The screenshot shows the simulator after the program has been loaded and executed. The state is as follows:

- Registers:**
 - GPR 0: '0000000000010101', LD button.
 - GPR 1: '0000000001000011', LD button.
 - GPR 2: '0000000000000011', LD button.
 - GPR 3: '0000000000000000', LD button.
 - IXR 1: '0000000000000000', LD button.
 - IXR 2: '0000000000000000', LD button.
 - IXR 3: '0000000000000000', LD button.
 - PC: '000100000100', LD button.
 - MAR: LD button.
 - MBR: LD button.
 - IR: '1100100100000001', LD button.
 - MFR: LD button.
 - Privileged: checkbox.
- Buttons:** 'Store', 'Load', 'init', 'Reset All', 'Program 1', 'SS', 'Run', 'HALT', 'RUN'.
- Keyboard:** Buttons 15 through 0.
- Console Printer:** Text area showing 'Out: 0000000001000011, Integer Value67'.
- Field Engineers Console:** A scrollable text area containing execution logs:


```

Instruction: 0000100000011000 OP Code:STR Performed at Address: 001000000000
Instruction: 0000010100011111 OP Code:LDR Performed at Address: 001000000000
Instruction: 0000100100011111 OP Code:STR Performed at Address: 001000000000
Instruction: 0001100000000001 OP Code:AIR Performed at Address: 001000000100
Instruction: 0011101100111100 OP Code:SOB Performed at Address: 001000000100
Instruction: 0010110000111101 OP Code:JMA Performed at Address: 001000000111
Instruction: 0000010100011111 OP Code:LDR Performed at Address: 000100000001
Instruction: 1100100100000001 OP Code:OUT Performed at Address: 000100000001
      
```
- Cache:** A scrollable text area containing memory data:


```

001000000001 0000100000011000
001000000000 0000100100010111
000000110001 0011110001110111
000000110000 0001011000011000
000000101111 0000100100011000
000000101011 0011110100111010
000000101010 0010000100111001
000000101001 0001010100111000
000000101000 0000100000011000
      
```

PHASE III

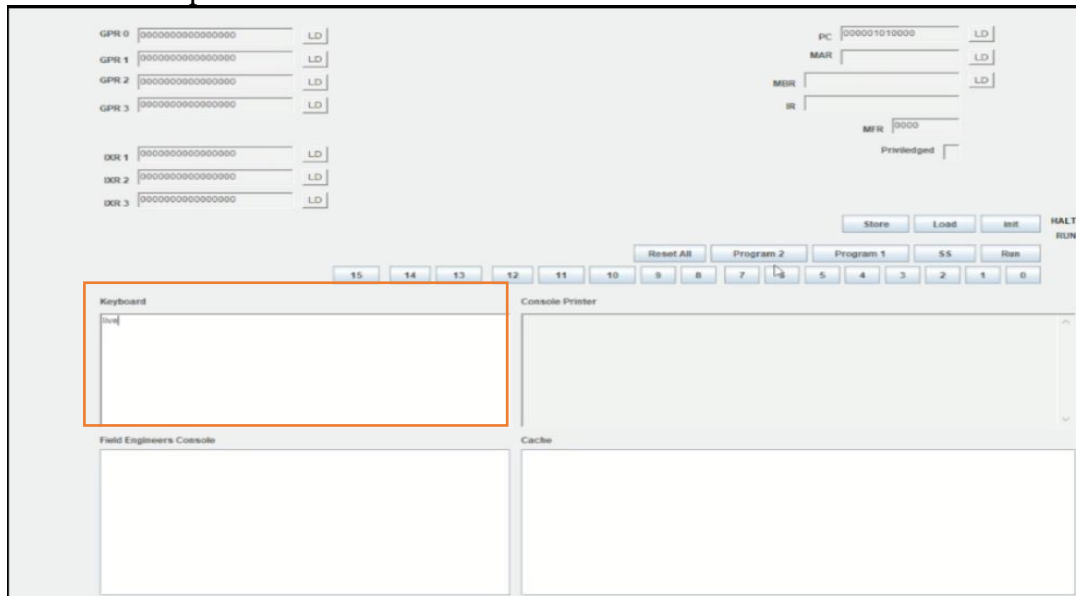
New segment added:

From the UI perspective, a new button (program 2) was added, which on click enables the user to test program 2 implementation.

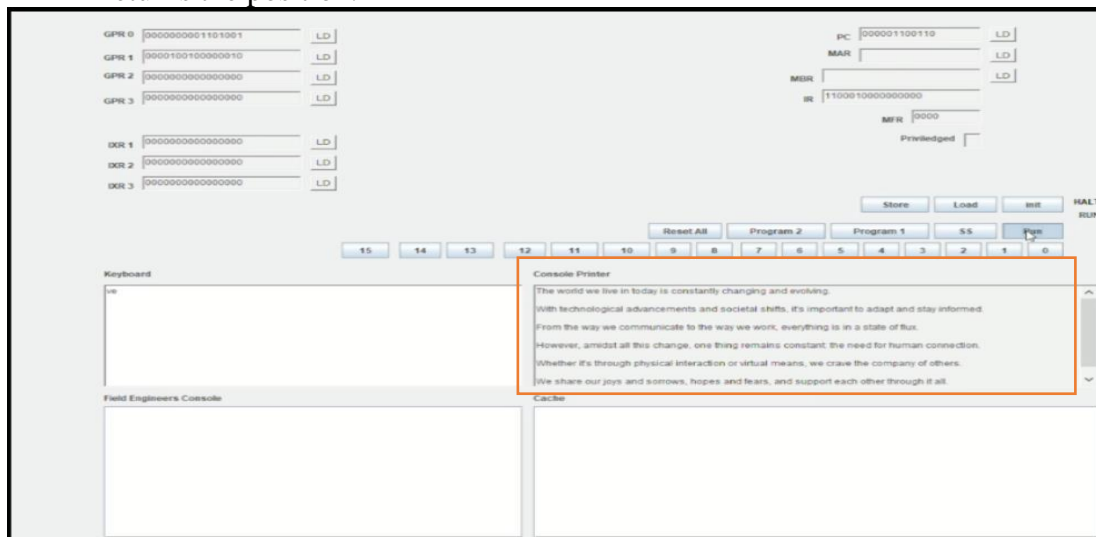


Steps to execute program 2:

1. Click on 'Reset all' button
2. Click on 'Program 2 button'.
3. Now, provide the input value, this is the value to be searched from the sentence/ words provided in the sentence.txt.



4. In the test case, we have used in the program, we have given 'live' as the word, and this word will be looked out in the sentence.txt file to see where it first occurs and returns the position.



The image shows a simulation interface for a computer architecture. At the top, there are several registers: GPR 0, GPR 1, GPR 2, GPR 3, DRR 1, DRR 2, and DRR 3, each with a value and a ".LD" button. To the right, there are PC, MAR, MDR, and IR registers, also with values and ".LD" buttons. Below these are buttons for "Store", "Load", and "int". A "HALT" button is on the far right. In the center, there are buttons for "Reset AR", "Program 2", "Program 1", "S5", and "Run". Below these are buttons for "15", "14", "13", "12", "11", "10", "9", "8", "7", "6", "5", "4", "3", "2", "1", and "0". On the left, there is a "Keyboard" section with a text input field. On the right, there is a "Console Printer" section with a text output field. Below the keyboard, there is a "Field Engineers Console" section with a list of instructions and their execution details. Below the console printer, there is a "Cache" section with a list of cache entries.

GPR 0: 000000000100000 .LD
GPR 1: 00000000001010 .LD
GPR 2: 000000000110000 .LD
GPR 3: 000000000110100 .LD
DRR 1: 000000000000000 .LD
DRR 2: 000000000000000 .LD
DRR 3: 000000000000000 .LD

PC: 011100010110 .LD
MAR: .LD
MDR: .LD
IR: 1100101100000001
MIR: 0000
Privileged: ☐

Store Load int HALT
RUN

Reset AR Program 2 Program 1 S5 Run

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Keyboard

Console Printer

With technological advancements and societal shifts, it's important to adapt and stay informed.
From the way we communicate to the way we work, everything is in a state of flux.
However, amidst all this change, one thing remains constant: the need for human connection.
We share our joys and sorrows, hopes and fears, and support each other through it all.
T.S.A.

Field Engineers Console

Instruction: 0000011000111001 OP Code: LDA Performed at Address: 011100010110
Instruction: 0100011000000000 OP Code: DCD Performed at Address: 01110001110
Instruction: 0001010000111111 OP Code: AIR Performed at Address: 01110001101
Instruction: 0001101000010001 OP Code: AIR Performed at Address: 01110001110
Instruction: 0000110000011111 OP Code: LDA Performed at Address: 01110001111
Instruction: 0001100000000001 OP Code: AIR Performed at Address: 01110001000
Instruction: 1100100000000001 OP Code: OUT Performed at Address: 01110001000
Instruction: 1100101000000001 OP Code: OUT Performed at Address: 01110001001
Instruction: 0001101100011111 OP Code: AIR Performed at Address: 01110001001
Instruction: 0001101100010001 OP Code: AIR Performed at Address: 01110001000
Instruction: 1100101100000001 OP Code: OUT Performed at Address: 01110001010

Cache

011100010001 1100100000000001
011100010000 0001100000000001
011100001111 0000110000011111
011100001110 0001101000010001
011100001101 0001101000011111
011100001100 0100011000000000
011100001011 0000011000111001
011100001010 0101000001000000
011100001001 0100000110000000
011100001000 0000111000001010
011100000111 0000110000000001
011100000110 0000101100011010

Traps and Machine faults:

OpCode8	Instruction	Description
0	HALT	Stops the machine
30	TRAP Code	Traps to memory address 0, which contains the address of a table in memory. Stores the PC+1 in memory location 2. The table can have a maximum of 16 entries representing 16 routines for user-specified instructions stored elsewhere in memory. Trap code contains an index into the table, e.g. it takes values 0 – 15. When a TRAP instruction is executed, it goes to the routine whose address is in memory location 0, executes those instructions, and returns to the instruction stored in memory location 2. The PC+1 of the TRAP instruction is stored in memory location 2.

1. In case of TRAP, PC+1 is stored at memory address 2 and then the PC loads routine stored at address 0000 and executes all instructions and then set program counter to content of address.
2. An erroneous condition in the machine will cause a machine fault. The machine traps to memory address 1, which contains the address of a routine to handle machine faults.

The possible machine faults that are predefined are:

ID	Fault
0	Illegal Memory Address to Reserved Locations MFR set to binary 0001.
1	Illegal TRAP code MFR set to binary 0002.
2	Illegal Operation Code MFR set to binary 0003.
3	Illegal Memory Address beyond 2048 (memory installed) MFR set to binary 0004.

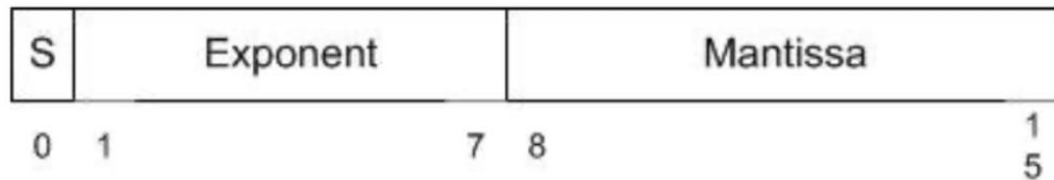
On occurrence of machine fault conditions, MFR field is set to corresponding binary decimal values. 0000 represents no fault.

PHASE IV

Floating point/Vector Operations:

In order to implement floating point instructions, we need two float registers FR0 and FR1. The register field in the 16-bit instruction denotes which floating register to use.

The content of float registers is binary value based on the below format:



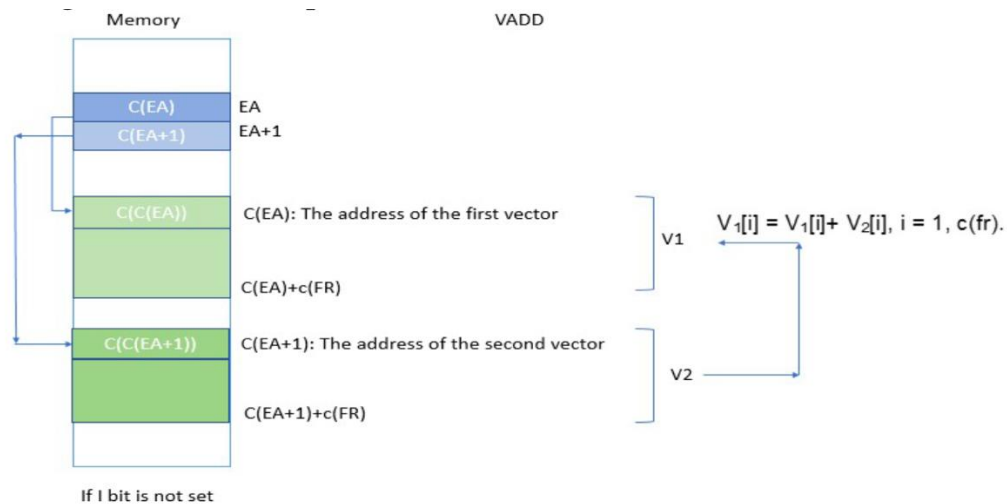
In the 16-bit binary, first bit denotes the sign of float value, second 8 bits denotes the exponent value and last 8 bits denotes the mantissa. Using sign, exponent and mantissa the actual float value can be evaluated.

OpCode	Instruction	Description
33	FADD fr, x, address[,I]	Floating Add Memory To Register $c(fr) \leftarrow c(fr) + c(EA)$ $c(fr) \leftarrow c(fr) + c(c(EA))$, if I bit set fr must be 0 or 1. OVERFLOW may be set
34	FSUB fr, x, address[,I]	Floating Subtract Memory From Register $c(fr) \leftarrow c(fr) - c(EA)$ $c(fr) \leftarrow c(fr) - c(c(EA))$, if I bit set fr must be 0 or 1 UNDERFLOW may be set

VADD:

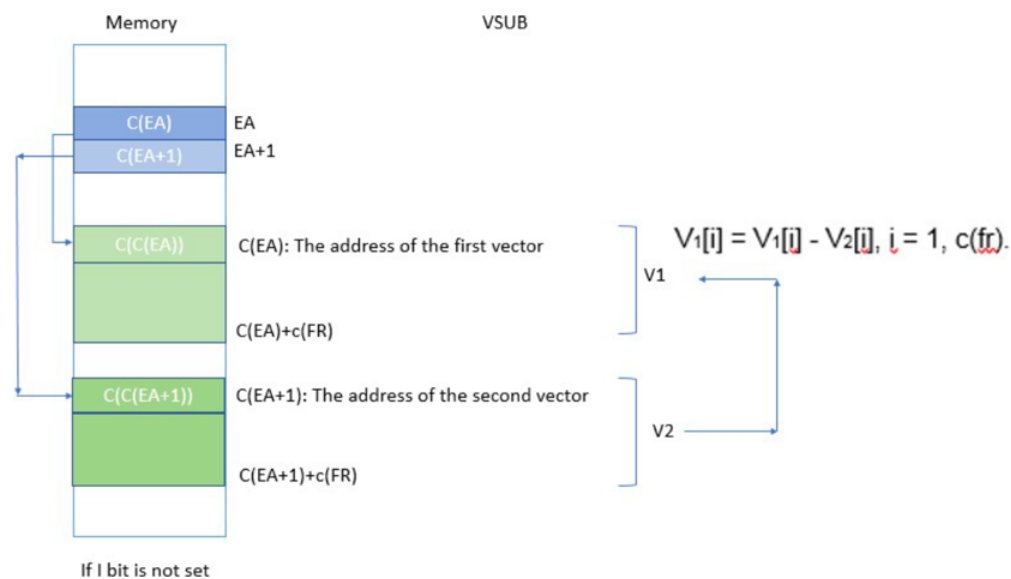
035	VADD fr, x, address[,I]	Vector Add fr contains the length of the vectors $c(EA)$ or $c(c(EA))$, if I bit set, is address of first vector $c(EA+1)$ or $c(c(EA+1))$, if I bit set, is address of the second vector. Let V_1 be vector at address; Let V_2 be vector at address+1 Then, $V_1[i] = V_1[i] + V_2[i]$, $i = 1, c(fr)$.
-----	-------------------------	--

The below diagram shows the implementation of VADD in the simulator.

**VSUB:**

036	VSUB fr, x, address[,I]	<p>Vector Subtract</p> <p>fr contains the length of the vectors</p> <p>c(EA) or c(c(EA)), if I bit set is address of first vector</p> <p>c(EA+1) or c(c(EA+1)), if I bit set is address of the second vector</p> <p>Let V_1 be vector at address; Let V_2 be vector at address+1</p> <p>Then, $V_1[i] = V_1[i] - V_2[i], i = 1, c(fr).$</p>
-----	-------------------------	--

The diagram below shows the implementation of VSUB:



Updated UI elements:

FR0, FR1: Floating point register, for floating point operations.

The screenshot displays a simulation interface for a computer system. At the top, there are input fields for registers: GPR 0, GPR 1, GPR 2, GPR 3, IXR 1, IXR 2, IXR 3, FR 0, and FR 1. Each field has an 'LD' button next to it. To the right, there are fields for PC, MAR, MBR, IR, and MFR, also with 'LD' buttons. A 'Privileged' checkbox is located below these. In the center, there is a row of buttons: 'Store', 'Load', 'init', 'Reset All', 'Program 2', 'Program 1', 'SS', and 'Run'. Below these are buttons for 'FADD Example', 'VADD Example', and 'CNVRT Example'. A row of 16 buttons, numbered 15 down to 0, is positioned below the example buttons. On the far right, there are 'HALT' and 'RUN' buttons. At the bottom, there are four output windows: 'Keyboard', 'Console Printer', 'Field Engineers Console', and 'Cache'.

Registers and Input Fields:

- GPR 0, GPR 1, GPR 2, GPR 3
- IXR 1, IXR 2, IXR 3
- FR 0, FR 1
- PC, MAR, MBR, IR, MFR
- Privileged ☐

Control Buttons:

- Store, Load, init
- Reset All, Program 2, Program 1, SS, Run
- FADD Example, VADD Example, CNVRT Example
- 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
- HALT, RUN

Output Windows:

- Keyboard
- Console Printer
- Field Engineers Console
- Cache

Execution Steps:

1. FADD fr, x, address[,I] Floating Add Memory To Register $c(fr) \leftarrow c(fr) + c(EA)$,
 $c(fr) \leftarrow c(fr) + c(c(EA))$, if I bit set fr must be 0 or 1. **OVERFLOW** may be set

Opcode: 33

- Set MAR to B:111 and MBR to B: 0000000000001000.
- Set MAR to B: 1000 and MBR to B: 0110110000000111.
- Press store to store MBR value at MAR address.
- Set PC to B: 1000
- Click SS
- The FR0 gets updated with the result.

The screenshot displays a simulation interface for a computer architecture. It includes several sections:

- Registers:**
 - GPRs (General Purpose Registers):** GPR 0, 1, 2, and 3, each with a value of 0000000000000000 and an LD button.
 - IXRs (Index Registers):** IXR 1, 2, and 3, each with a value of 0000000000000000 and an LD button.
 - FRs (Floating Point Registers):** FR 0 (000000000001000) and FR 1 (0000000000000000), each with an LD button.
 - PC (Program Counter):** 000000100001 with an LD button.
 - MAR (Memory Address Register):** Empty with an LD button.
 - MBR (Memory Buffer Register):** Empty with an LD button.
 - IR (Instruction Register):** 0110110000000111.
 - MFR (Memory Format Register):** 0000.
 - Privileged:** A checkbox that is currently unchecked.
- Execution Controls:**
 - Buttons: Store, Load, init, HALT, RUN.
 - Buttons: Reset All, Program 2, Program 1, SS, Run.
 - A row of buttons numbered 15 down to 0, with button 3 highlighted in red.
- Examples:**
 - FADD Example
 - VADD Example
 - CNVRT Example
- Keyboard:** A large empty text area for input.
- Console Printer:** A large empty text area for output.
- Field Engineers Console:**
 - Instruction Store: At Memory:000000000111 Value:0000000000001000
 - Instruction Store: At Memory:000000100000 Value:0110110000000111
 - Instruction: 0110110000000111 OP Code:FADD Performed at Address: 0000001000
- Cache:**
 - 000000100000 0110110000000111
 - 000000100000 0110110000000111
 - 000000000111 0000000000001000

2. FSUB fr, x, address[,I] Floating Subtract Memory From Register $c(fr) \leftarrow c(fr) - c(EA)$, $c(fr) \leftarrow c(fr) - c(c(EA))$, if I bit set fr must be 0 or 1. UNDERFLOW may be set

Opcode: 34

i. Set MAR to 111 and MBR to 0000000000001000 and click Store.

ii. Set MAR to 1000 and MBR to 0111000000000111 and click

Store.

iii. Set PC to 1000

iv. Load FR0 with 100

v. Click SS

The screenshot displays a computer architecture simulation interface. At the top, there are input fields for various registers and memory locations, each with a 'LD' button. On the left, GPR 0 through GPR 3, IXR 1 through IXR 3, and FR 0 through FR 1 are listed. On the right, PC, MAR, MBR, IR, and MFR are listed. Below these are buttons for 'Store', 'Load', 'Init', 'Reset All', 'Program 2', 'Program 1', 'SS', and 'Run'. A row of buttons numbered 15 down to 0 is visible, with button 2 highlighted in red. Below the register fields are buttons for 'FADD Example', 'VADD Example', and 'CNVRT Example'. The bottom section contains a 'Keyboard' input area, a 'Console Printer' output area, and a 'Field Engineers Console' log showing instruction load and store details. To the right of the log is a 'Cache' section displaying memory addresses and values.

Registers and Memory:

- GPR 0: 0000000000000000 LD
- GPR 1: 0000000000000000 LD
- GPR 2: 0000000000000000 LD
- GPR 3: 0000000000000000 LD
- IXR 1: 0000000000000000 LD
- IXR 2: 0000000000000000 LD
- IXR 3: 0000000000000000 LD
- FR 0: 0000000000000100 LD
- FR 1: 0000000000000000 LD
- PC: 000000001001 LD
- MAR: 000000001000 LD
- MBR: 0111000000000111 LD
- IR: 0111000000000111
- MFR: 0000
- Privileged: ☐

Execution Controls:

- Store, Load, Init, Reset All, Program 2, Program 1, SS, Run
- Buttons 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2 (highlighted), 1, 0
- FADD Example, VADD Example, CNVRT Example

Field Engineers Console:

```

Instruction Load: At Memory:000000000111 Value:0000000000000000
Instruction Store: At Memory:000000001000 Value:0111000000000111
Instruction: 0111000000000111 OP Code:FSUB Performed at Address: 000000001000
  
```

Cache:

```

000000001000 0111000000000111
000000001000 0111000000000111
000000000111 0000000000000000
  
```

3. VADD fr, x, address[I], Vector Add

fr contains the length of the vectors

c(EA) or **c(c(EA))**, if **I** bit set, is address of first vector

c(EA+1) or **c(c(EA+1))**, if **I** bit set, is address of the second vector

Let **V1** be vector at address; Let **V2** be vector at address+1, Then, $V1[i] = V1[i] + V2[i]$, $i = 1, c(fr)$.

Opcode: 35

- i. Set MAR to B:111 and MBR to B:00000000000010100.
- ii. Set MAR to B: 1000 and MBR to B: 0000000000000101.
- iii. Set MAR to B: 1001 and MBR to B: 0000000000000010.
- iv. Set MAR to B: 1010 and MBR to B: 0000000000000001.
- v. Set MAR to B: 110 and MBR to B: 0111011100000111.
- vi. Press store to store MBR value at MAR address.
- vii. Set PC to B: 110
- viii. Click SS
- ix. The memory gets updated with the result.

The screenshot displays the Field Engineers Console (FEC) interface. The top section shows various registers and instruction fields:

- GPR 0-3:** All set to 0000000000000000.
- IXR 1-3:** All set to 0000000000000000.
- FR 0-1:** Both set to 0000000000000000.
- PC:** 000000100001
- MAR:** (Empty)
- MBR:** (Empty)
- IR:** 0111011100000111
- MFR:** 0000
- Privileged:** (Unchecked)

Below the registers, there are buttons for **Store**, **Load**, **init**, **Reset All**, **Program 2**, **Program 1**, **SS**, and **Run**. A row of buttons labeled 15 through 0 is visible, with button 4 highlighted in red.

The bottom section contains the **Field Engineers Console** and **Cache** windows:

- Field Engineers Console:**

```

Instruction Store: At Memory:000000000111 Value:00000000000010100
Instruction Store: At Memory:000000001000 Value:0000000000000101
Instruction Store: At Memory:000000100000 Value:0111011100000111
Instruction: 0111011100000111 OP Code:VADD Performed at Address: 0000001000

```
- Cache:**

```

000000100000 0111011100000111
000000100000 0111011100000111
000000001000 0000000000000101
000000000111 0000000000001010

```


4. VSUB fr, x, address[,I], Vector Subtract**fr contains the length of the vectors****c(EA) or c(c(EA)), if I bit set is address of first vector****c(EA+1) or c(c(EA+1)), if I bit set is address of the second vector****Let V1 be vector at address; Let V2 be vector at address+1, Then, $V1[i] = V1[i] - V2[i]$, $i = 1, c(fr)$.****Opcode: 36**

- i. Set MAR to B:111 and MBR to B:00000000000010100.
- ii. Set MAR to B: 1000 and MBR to B: 0000000000000101.
- iii. Set MAR to B: 1001 and MBR to B: 0000000000000010.
- iv. Set MAR to B: 1010 and MBR to B: 0000000000000001.
- v. Set MAR to B: 110 and MBR to B: 0111101100000111.
- vi. Press store to store MBR value at MAR address.
- vii. Set PC to B: 110
- viii. Click SS
- ix. The memory gets updated with the result.

The screenshot displays a computer architecture simulator interface. At the top, there are input fields for registers: GPR 0 through GPR 3, all set to 0000000000000000, and LD buttons. Below these are IXR 1 through IXR 3, also set to 0000000000000000, with LD buttons. Further down are FR 0 and FR 1, set to 0000000000000000, with LD buttons. On the right side, there are fields for PC (000000000111), MAR (000000000110), MBR (0111101100000111), and IR (0111101100000111), each with an LD button. Below these is an MFR field set to 0000 and a Privileged checkbox. A row of buttons includes Store, Load, init, and HALT RUN. Below this is a row of buttons: Reset All, Program 2, Program 1, SS, and Run. A numeric keypad with buttons 15 through 0 is present, with buttons 2 and 1 highlighted in red. Below the keypad are buttons for FADD Example, VADD Example, and CNVRT Example. The bottom section contains a Keyboard input area and a Console Printer output area. At the very bottom, there are two scrollable text areas: 'Field Engineers Console' and 'Cache'. The 'Field Engineers Console' shows instruction store details and the executed instruction: 'Instruction: 0111101100000111 OP Code:VSUB Performed at Address: 0000000001'. The 'Cache' area shows a memory layout with addresses and values.

5. CNVRT r, x, address[,I], Convert to Fixed/FloatingPoint:

If F = 0, convert c(EA) to a fixed-point number and store in r. If F = 1, convert c(EA) to a floating-point number and store in FR0. The r register contains the value of F before the instruction is executed.

Opcode: 37

- i. Set GPR0 as B: 0001
- ii. Set MAR to B:111 and MBR to B: 0000000000011111.
- iii. Set MAR to B: 1000 and MBR to B: 0111110000000111.
- iv. Press store to store MBR value at MAR address.
- v. Set PC to B: 1000
- vi. Click SS
- vii. The FR0 gets updated with the converted value.

The screenshot displays a simulation window with the following components:

- Registers:**
 - GPR 0: 0000000000000001 (LD)
 - GPR 1: 0000000000000000 (LD)
 - GPR 2: 0000000000000000 (LD)
 - GPR 3: 0000000000000000 (LD)
 - IXR 1: 0000000000000000 (LD)
 - IXR 2: 0000000000000000 (LD)
 - IXR 3: 0000000000000000 (LD)
 - FR 0: 000010111111000 (LD)
 - FR 1: 0000000000000000 (LD)
- Control and Status:**
 - PC: 00000100001 (LD)
 - MAR: (LD)
 - MBR: (LD)
 - IR: 0111110000000111
 - MFR: 0000
 - Privileged: ☐
- Execution Controls:**
 - Buttons: Store, Load, init, HALT, RUN
 - Buttons: Reset All, Program 2, Program 1, SS, Run
 - Memory Address Buttons: 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
 - Buttons: FADD Example, VADD Example, CNVRT Example
- Field Engineers Console:**

```

Instruction Store: At Memory:000000000111 Value:0000000000011111
Instruction Store: At Memory:00000100000 Value:0111110000000111
Instruction: 0111110000000111 OP Code:CNVRT Performed at Address: 00000100000
          
```
- Cache:**

```

000000100000 0111110000000111
000000100000 0111110000000111
000000000111 0000000000011111
          
```
- Keyboard and Console Printer:** Empty text areas for input and output.

6. LDFR fr, x, address [i], Load Floating Register From Memory, fr = 0..1 fr <- c(EA), c(EA+1)

fr <- c(c(EA), c(EA)+1), if I bit set

Opcode: 50

- Set MAR to B:111 and MBR to B: 0000000000001000).
- Set MAR to B: 1000 and MBR to B: 0000000000001001.
- Set MAR to B: 110 and MBR to B: 1010000000000111.
- Press store to store MBR value at MAR address.
- Set PC to B: 110
- Click SS
- The FR0 gets updated with the value.

The screenshot displays a computer architecture simulator interface. At the top, there are input fields for various registers and memory locations, each with a 'LD' button. On the left, GPR 0-3 and IXR 1-3 are set to 0000000000000000. On the right, PC is 00000000111, MAR is 00000000110, MBR is 1010000000000111, and IR is 1010000000000111. Below these, FR 0 is 000000000001001 and FR 1 is 0000000000000000. A 'Privileged' checkbox is unchecked. In the center, there are buttons for 'Store', 'Load', 'init', 'Reset All', 'Program 2', 'Program 1', 'SS', and 'Run'. Below these are buttons for 'FADD Example', 'VADD Example', and 'CNVRT Example'. A row of buttons numbered 15 down to 0 is visible, with 15, 14, and 13 highlighted in red. At the bottom, there are two large text areas labeled 'Keyboard' and 'Console Printer'. The 'Field Engineers Console' at the bottom left shows a log of instructions and their execution. The 'Cache' at the bottom right shows a list of memory addresses and their values.

Registers and Memory:

- GPR 0: 0000000000000000
- GPR 1: 0000000000000000
- GPR 2: 0000000000000000
- GPR 3: 0000000000000000
- IXR 1: 0000000000000000
- IXR 2: 0000000000000000
- IXR 3: 0000000000000000
- FR 0: 000000000001001
- FR 1: 0000000000000000
- PC: 00000000111
- MAR: 00000000110
- MBR: 1010000000000111
- IR: 1010000000000111
- MFR: 0000
- Privileged: ☐

Execution Controls:

- Store
- Load
- init
- Reset All
- Program 2
- Program 1
- SS
- Run

Examples:

- FADD Example
- VADD Example
- CNVRT Example

Console Log:

```

Instruction Store: At Memory:00000000111 Value:0000000000001000
Instruction Store: At Memory:000000001000 Value:0000000000001001
Instruction Store: At Memory:00000000110 Value:1010000000000111
Instruction: 1010000000000111 OP Code:LDFR Performed at Address: 0000000001
  
```

Cache:

```

000000000110 1010000000000111
000000000110 1010000000000111
0000000001000 0000000000001001
000000000111 0000000000001000
  
```

7. STFR fr, x, address [,i], Store Floating Register To Memory, fr = 0..1 EA, EA+1<-c(fr), c(EA), c(EA)+1 <- c(fr), if I-bit set

- i. Set MAR to 110 and MBR to B: 1010010000000111.
- ii. set FR0 to B: 1010101010101010
- iii. set PC to B: 110
- iv. Address 111 becomes 1010101010101010
- v. Address 1000 becomes 0000000000000000.

The screenshot displays a simulation interface for a computer architecture. It includes several sections:

- Registers:**
 - GPRs (General Purpose Registers):** GPR 0, 1, 2, and 3, each with a value of 0000000000000000 and an LD button.
 - IXRs (Index Registers):** IXR 1, 2, and 3, each with a value of 0000000000000000 and an LD button.
 - FRs (Floating Point Registers):** FR 0 contains 1010101010101010 and FR 1 contains 0000000000000000, both with LD buttons.
 - PC (Program Counter):** 000000000111 with an LD button.
 - MAR (Memory Address Register):** 000000001000 with an LD button.
 - MBR (Memory Buffer Register):** 0000000000000000 with an LD button.
 - IR (Instruction Register):** 1010010000000111 with an LD button.
 - MFR (Memory Format Register):** 0000 with an LD button.
 - Privileged:** A checkbox that is currently unchecked.
- Execution Controls:**
 - Buttons: Store, Load, init, HALT, RUN.
 - Buttons: Reset All, Program 2, Program 1, SS, Run.
 - A row of buttons numbered 15 down to 0, with button 3 highlighted in red.
 - Buttons: FADD Example, VADD Example, CNVRT Example.
- Keyboard:** A large empty text area for input.
- Console Printer:** A large empty text area for output.
- Field Engineers Console:**
 - Instruction Store: At Memory:000000000110 Value:1010010000000111
 - Instruction: 1010010000000111 OP Code:STFR Performed at Address: 0000000001
 - Instruction Load: At Memory:000000000111 Value:1010101010101010
 - Instruction Load: At Memory:000000001000 Value:0000000000000000
- Cache:**
 - 000000001000 0000000000000000
 - 000000000111 1010101010101010
 - 000000000110 1010010000000111
 - 000000000110 1010010000000111

The interface displays various components of a computer system, including registers, control unit, and memory.

Registers:

- GPR 0: 0000000000000000 LD
- GPR 1: 0000000000000000 LD
- GPR 2: 0000000000000000 LD
- GPR 3: 0000000000000000 LD
- IXR 1: 0000000000000000 LD
- IXR 2: 0000000000000000 LD
- IXR 3: 0000000000000000 LD
- FR 0: 1010101010101010 LD
- FR 1: 0000000000000000 LD

Control Unit:

- PC: 000000000111 LD
- MAR: 000000000111 LD
- MBR: 1010101010101010 LD
- IR: 1010010000000111
- MFR: 0000
- Privileged: ☐

Buttons:

- Store, Load, init, HALT, RUN
- Reset All, Program 2, Program 1, SS, Run
- FADD Example, VADD Example, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
- CNVRT Example

Keyboard:

Console Printer

Field Engineers Console:

```
Instruction Load: At Memory:000000000111 Value:1010101010101010
Instruction Load: At Memory:000000001000 Value:0000000000000000
Instruction Load: At Memory:000000001000 Value:0000000000000000
Instruction Load: At Memory:000000000111 Value:1010101010101010
```

Cache:

```
000000001000 0000000000000000
000000001000 0000000000000000
000000000111 1010101010101010
000000000110 1010010000000111
000000000110 1010010000000111
```

FADD Example button: Does exactly as shown in FADD instruction shown above, video attached in zip.

VADD Example button: Does exactly as shown in VADD instruction shown above, video attached in zip.

CNVRT Example button: Does exactly as shown in CNVRT instruction shown above, video attached in zip.