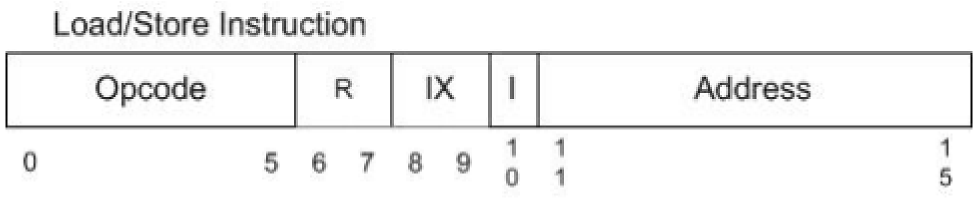
DESIGN DOCUMENT

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:

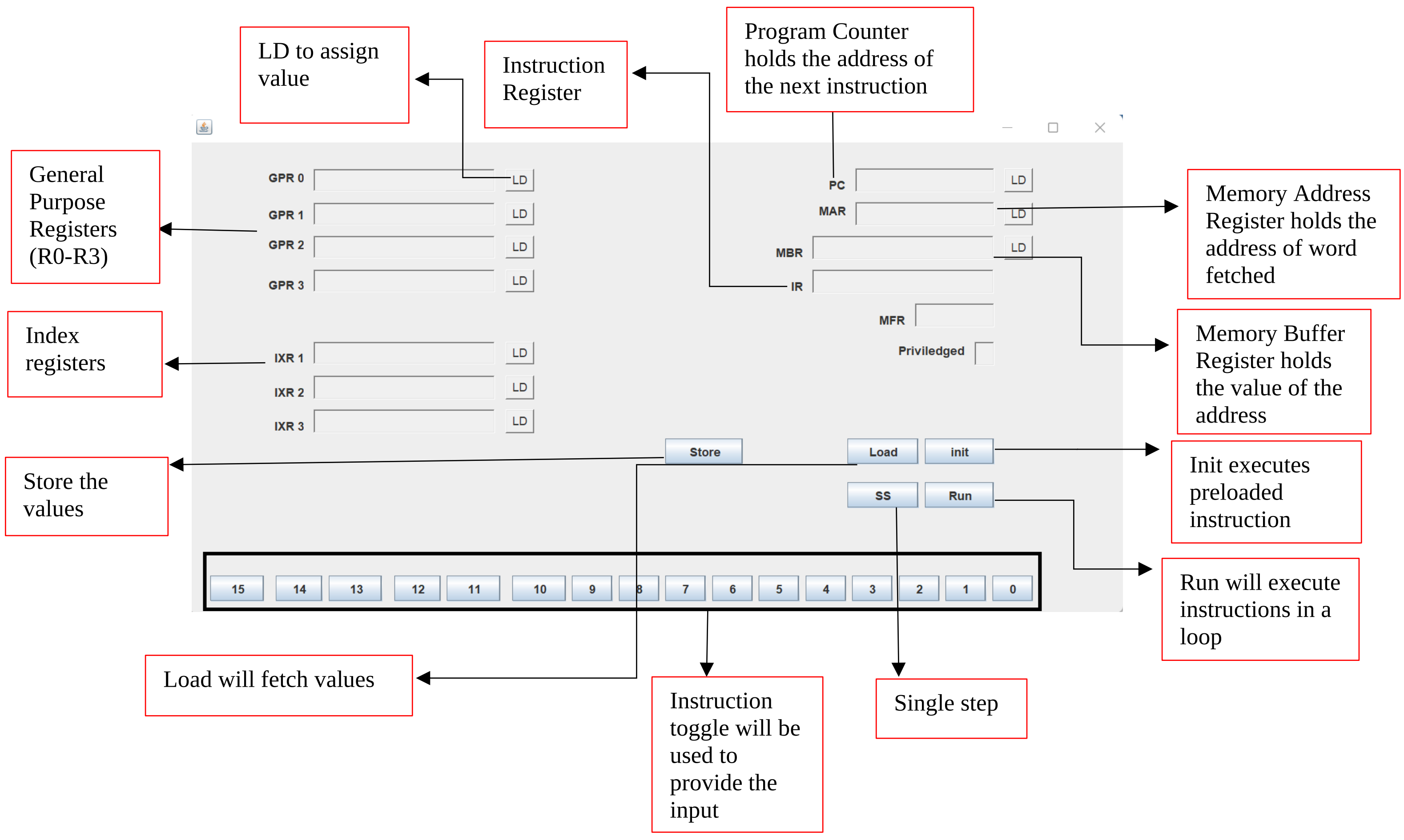
IX = 00: content (content (address field))

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):

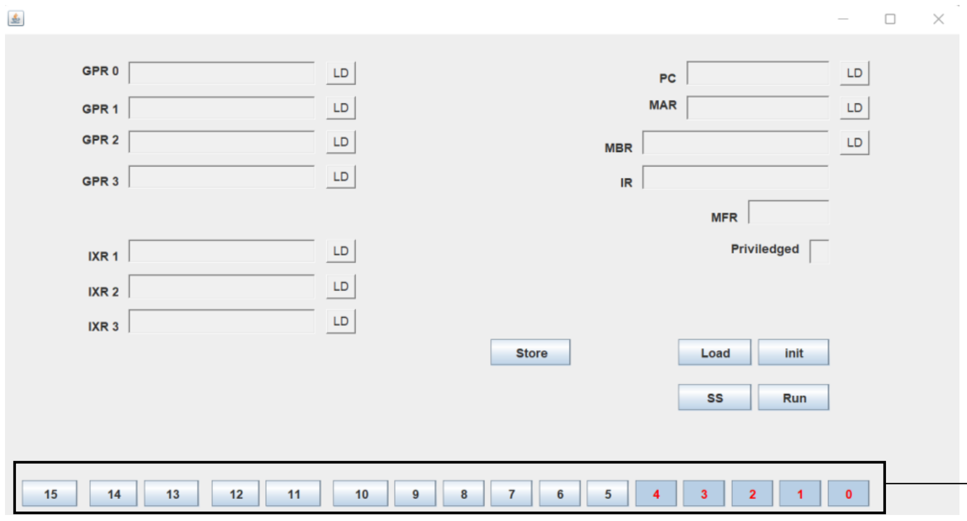
OpCode	Instruction	Description
01	LDR r x I address	Load Register From Memory, r = 0..3 r <- c(EA)
02	STR r x I address	Store Register To Memory, r = 0..3 Memory(EA) <- c(r)
03	LDA r x I address	Load Register with Address, r = 0..3 r <- EA
41	LDX x I address	Load Index Register from Memory, x = 1..3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

Simulator Design:


LD to assign value
Instruction Register
Program Counter holds the address of the next instruction
General Purpose Registers (R0-R3)
Memory Address Register holds the address of word fetched
Index registers
Memory Buffer Register holds the value of the address
Store the values
Init executes preloaded instruction
Run will execute instructions in a loop
Load will fetch values
Instruction toggle will be used to provide the input
Single step
GPR 0
GPR 1
GPR 2
GPR 3
IXR 1
IXR 2
IXR 3
LD
PC
MAR
MBR
IR
MFR
Priviledged
Store
Load
init
SS
Run
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

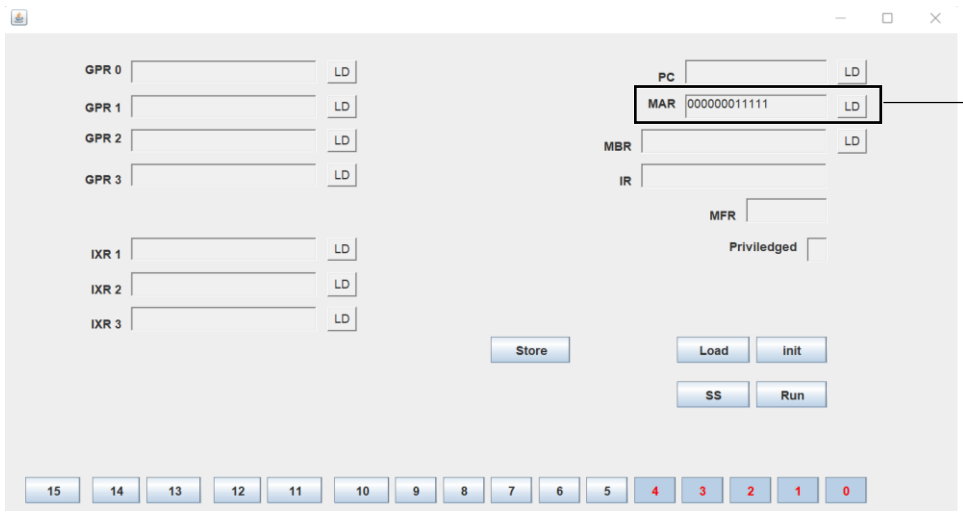
Steps to add instruction to memory:

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

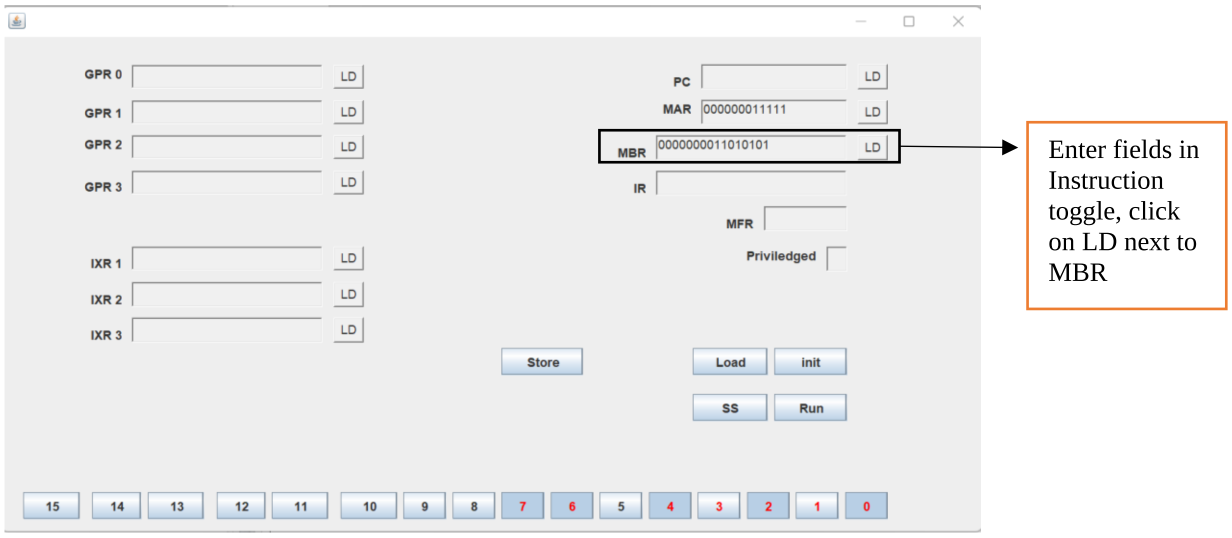
2. Click the “LD” button next to MAR to Load it to the MAR register.



MAR 000000011111

LD Button next to MAR to assign value to MAR

3. Do the above steps for MBR also:



MAR: 000000011111

MBR: 0000000011010101

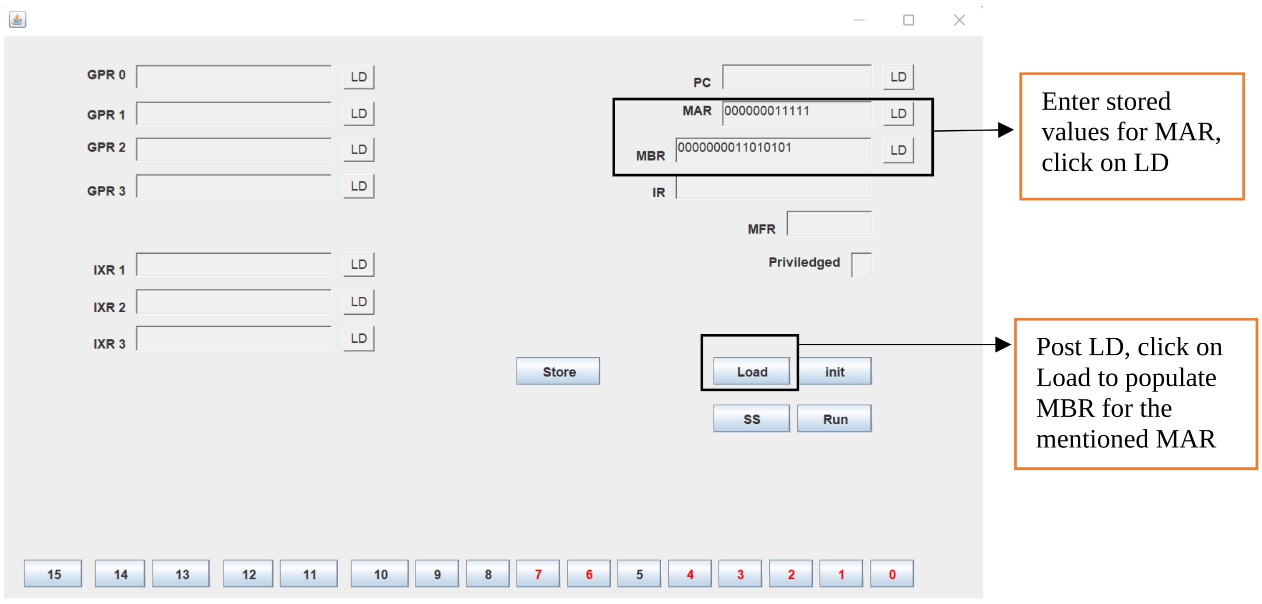
Enter fields in Instruction toggle, click on LD next to MBR

4. Click the "Store" button to store the MBR instruction entered to the MAR memory location

Steps to load instruction from the memory:

5. Enter fields under instruction toggle, click on "LD" next to MAR, this will populate fields in MAR

6. Now, click on "Load" button, this will populate MBR field for the corresponding MAR entered earlier



MAR: 000000011111

MBR: 0000000011010101

Enter stored values for MAR, click on LD

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

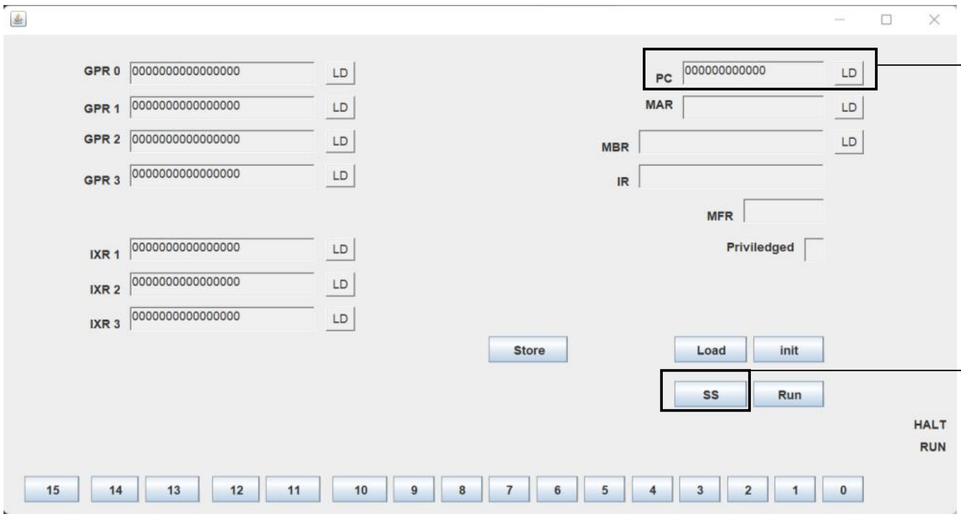
Steps to load IPL/ init file:

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

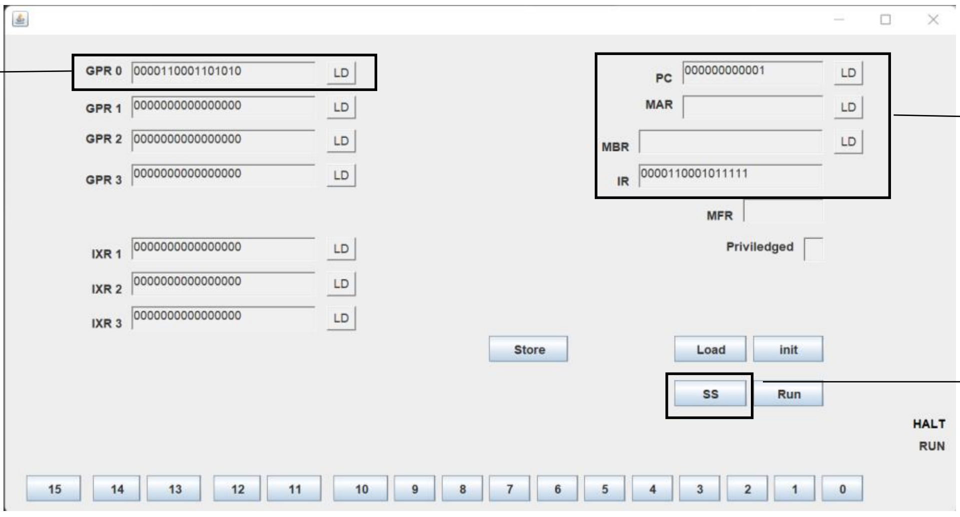


GPR 0: 0000000000000000
GPR 1: 0000000000000000
GPR 2: 0000000000000000
GPR 3: 0000000000000000
IXR 1: 0000000000000000
IXR 2: 0000000000000000
IXR 3: 0000000000000000
PC: 000000000000
MAR:

PC value populated once init is clicked

Click on SS to perform single step

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.



GPR 0: 0000110001101010
GPR 1: 0000000000000000
GPR 2: 0000000000000000
GPR 3: 0000000000000000
IXR 1: 0000000000000000
IXR 2: 0000000000000000
IXR 3: 0000000000000000
PC: 000000000001
MAR:
MBR:
IR: 0000110001011111

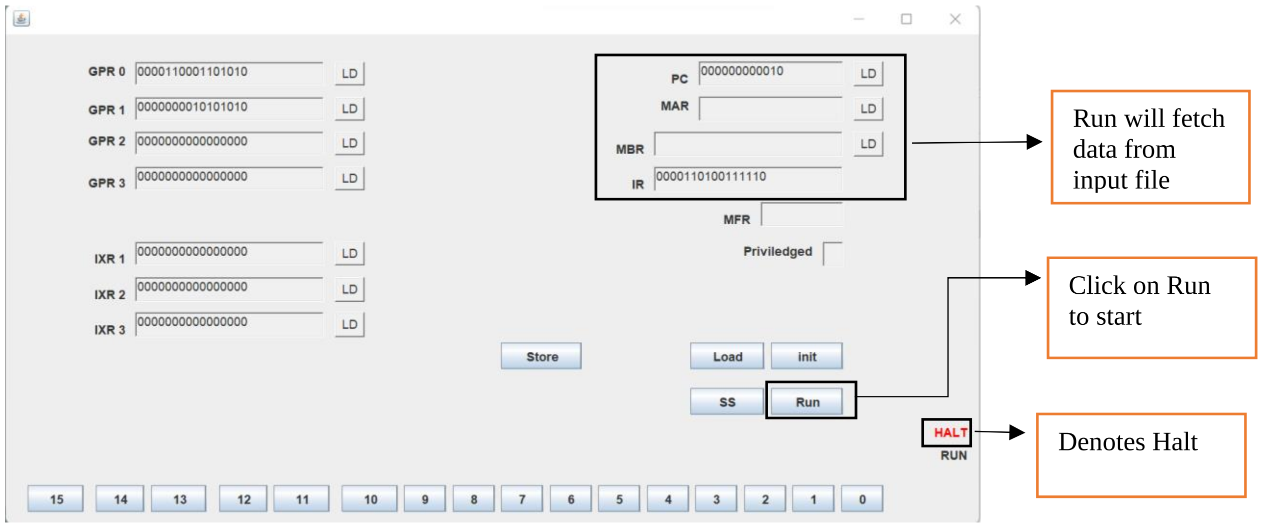
GPR value populated

Second SS. PC value incremented

Click on SS

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.



GPR 0 0000110001101010
GPR 1 0000000010101010
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000000010
MAR
MBR
IR 0000110100111110
MFR
Priviledged

Run will fetch data from input file

Click on Run to start

Denotes Halt

Phase II

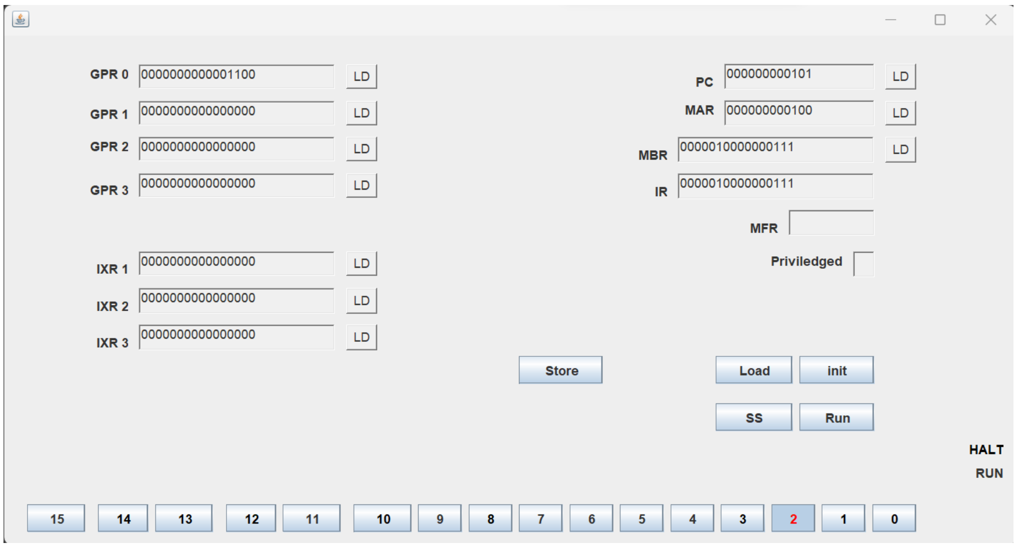
Updated UI elements:

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 <– 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.

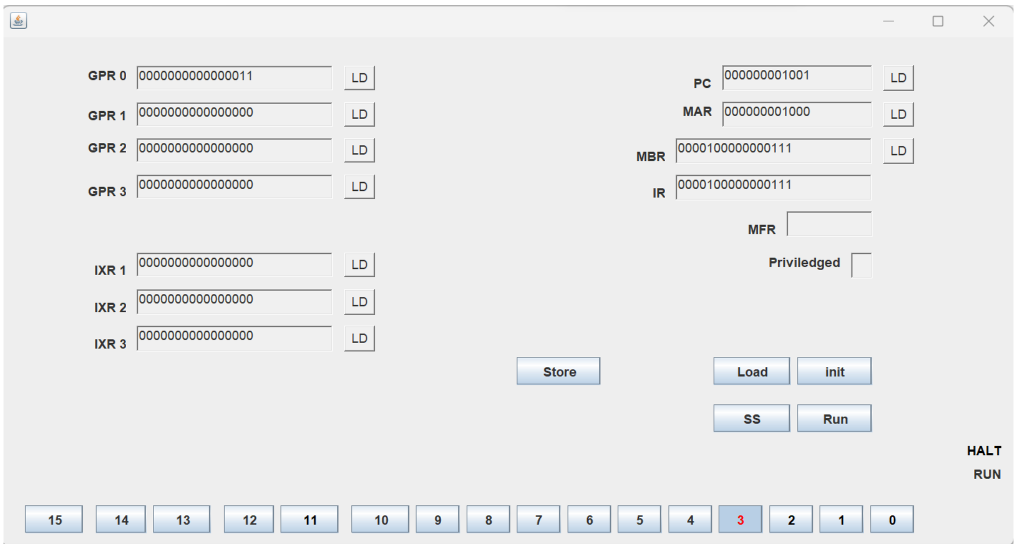


GPR 0 0000000000001100
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000000101
MAR 000000000100
MBR 0000010000000111
IR 0000010000000111
MFR
Priviledged

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

Opcode: 02

i. Load the GPR0 with B: 0000000000000011
ii. Load MBR with B: 000010000000111, with Opcode being 2
iii. Load MAR with B: 1000
iv. Set PC to B:1000
v. Click on store
vi. Click SS

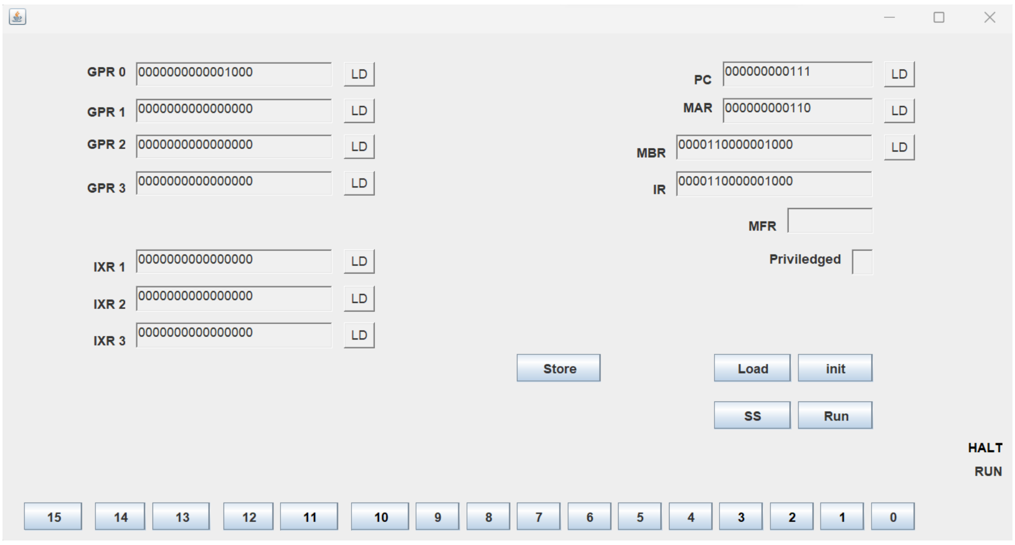


GPR 0 0000000000000011
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000001001
MAR 000000001000
MBR 0000100000000111
IR 0000100000000111
MFR
Priviledged

2. LDA r, x, address[,I] Load Register with Address, r = 0..3 r <− EA

Opcode: 03

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

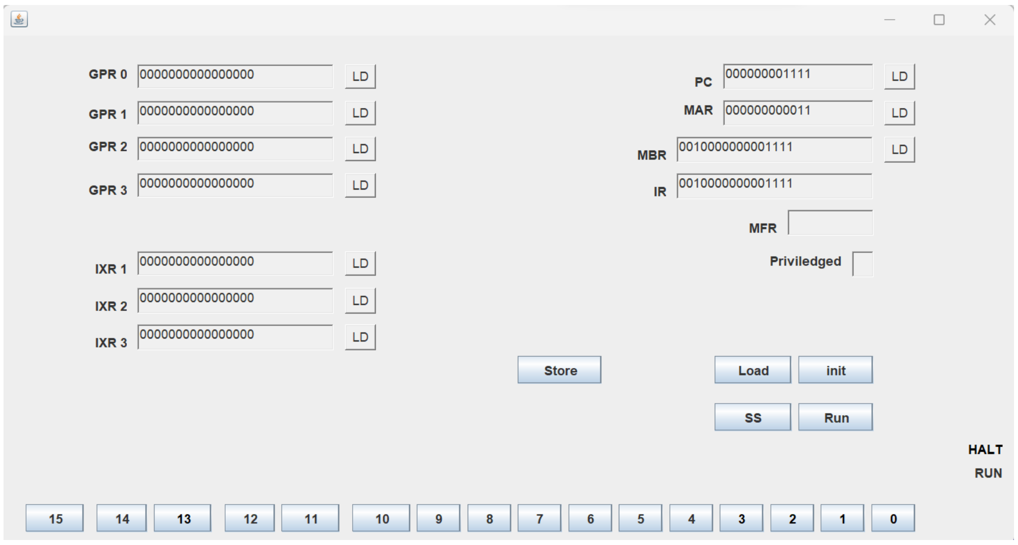


GPR 0	0000000000001000
GPR 1	0000000000000000
GPR 2	0000000000000000
GPR 3	0000000000000000
IXR 1	0000000000000000
IXR 2	0000000000000000
IXR 3	0000000000000000
PC	000000000111
MAR	000000000110
MBR	0000110000001000
IR	0000110000001000
MFR	
Priviledged	

5. JZ r, x, address[,I] Jump If Zero: If c(r) = 0, then PC <− EA Else PC <- 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

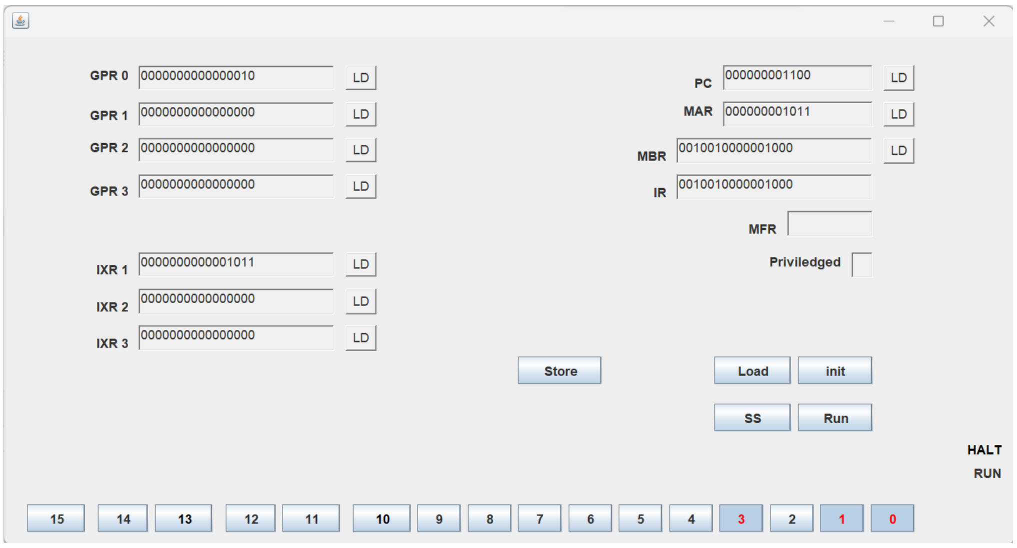


GPR 0 0000000000000000
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000001111
MAR 000000000011
MBR 0010000000001111
IR 0010000000001111
MFR
Priviledged

6. JNE r, x, address[,I] Jump If Not Equal: If c(r) != 0, then PC <–- EA Else PC <- PC +1

Opcode: 11

i. Load GPR0 with B: 0010
ii. Load MBR with B: 0010010000001000
iii. Load MAR with B: 1011
iv. Load PC with B: 1011
v. Click on store
vi. Click SS
vii. PC is Updated as c(r) != 0, then PC <– EA

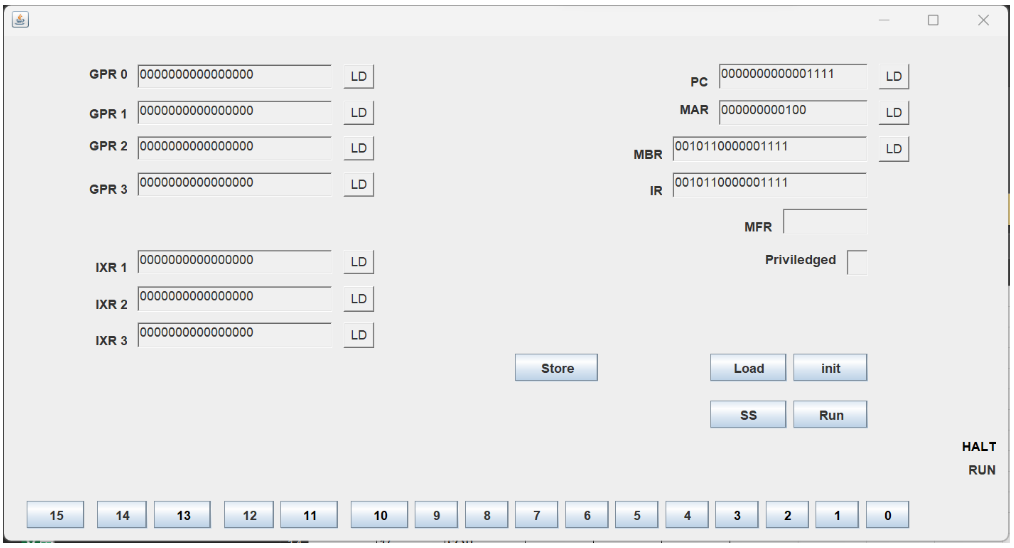


GPR 0 0000000000000010
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000001011
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000001100
MAR 000000001011
MBR 0010010000001000
IR 0010010000001000
MFR
Priviledged

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

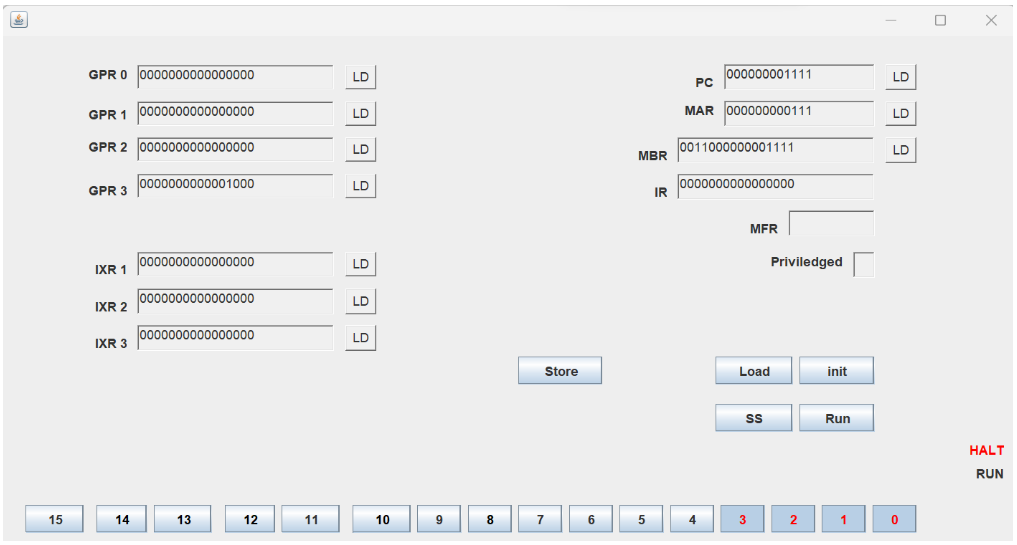


GPR 0 0000000000000000
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 0000000000001111
MAR 000000000100
MBR 0010110000001111
IR 0010110000001111
MFR
Priviledged

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

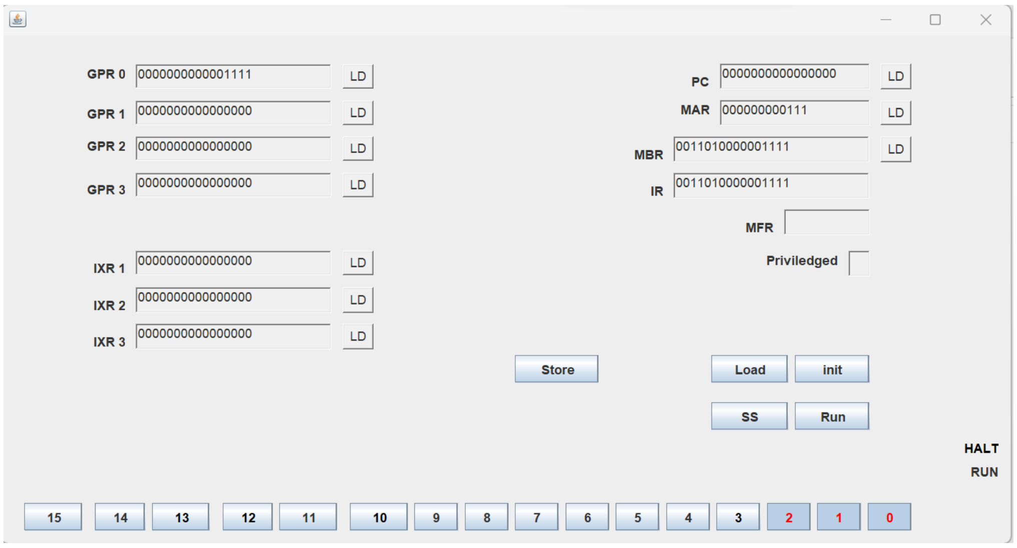
i. Load MBR with B: 0011000000001111
ii. Load MAR with B: 0111
iii. Set PC to B: 0111
iv. Click on store
v. Click SS
vi. PC is Updated as B:1111 and GPR3 becomes PC+1, i.e., B: 1000


GPR 0 0000000000000000
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000001000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000001111
MAR 000000000111
MBR 0011000000001111
IR 0000000000000000
MFR
Priviledged

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

Opcode: 15

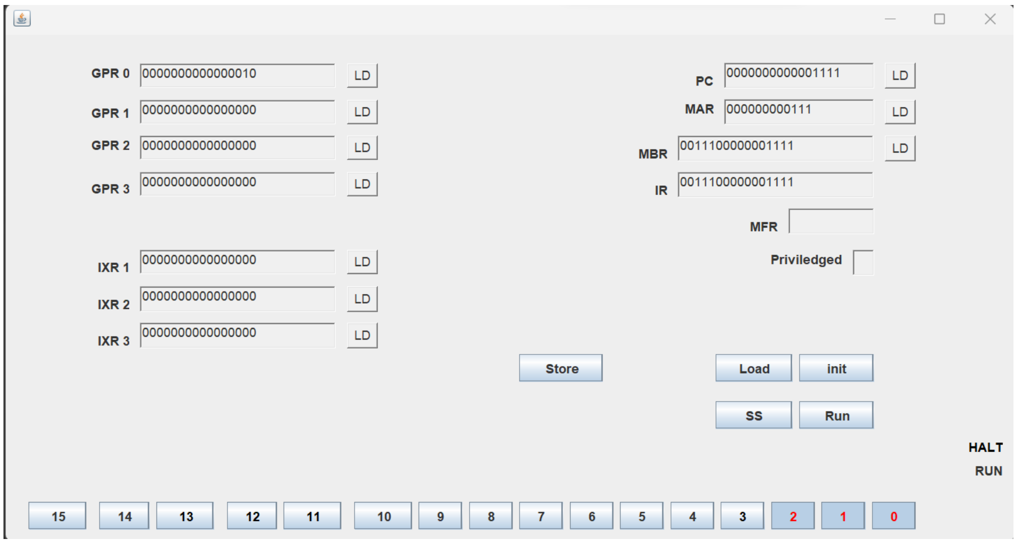
i. Load GPR3 with B: 0000
ii. Load MBR with B: 0011010000001111
iii. Load MAR with B: 0111
iv. Load PC with B: 0111
v. Click on store
vi. Click SS
vii. GPR0 becomes B:1111, which is EA, PC becomes B:0000 which is content of GP3.


GPR 0 0000000000001111
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 0000000000000000
MAR 000000000111
MBR 0011010000001111
IR 0011010000001111
MFR
Priviledged

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

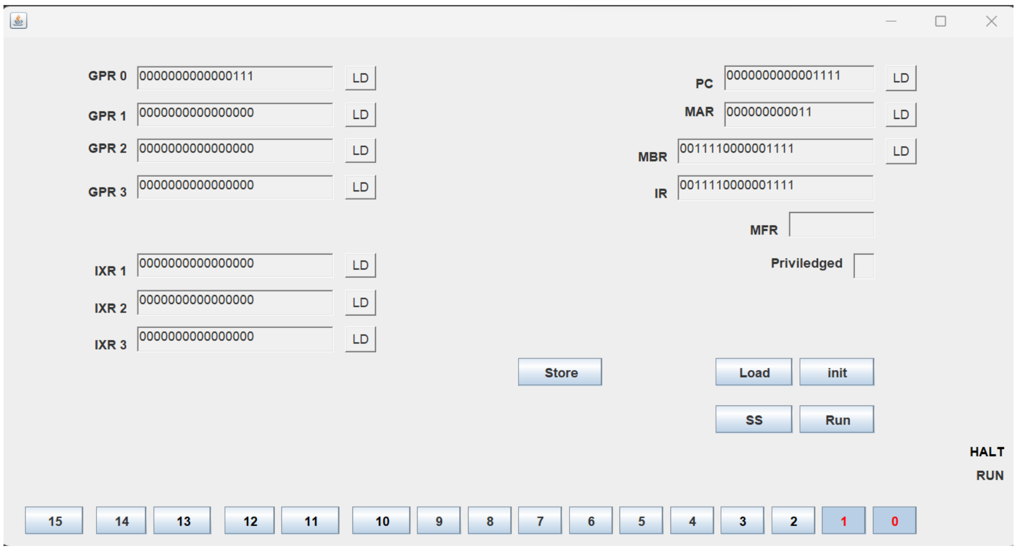


GPR 0 0000000000000010
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 0000000000001111
MAR 000000000111
MBR 0011100000001111
IR 0011100000001111
MFR
Priviledged

12. JGE r,x, address[,I] Jump Greater Than or Equal To: If c(r) >= 0, then PC <- EA Else PC <- PC + 1

Opcode: 17

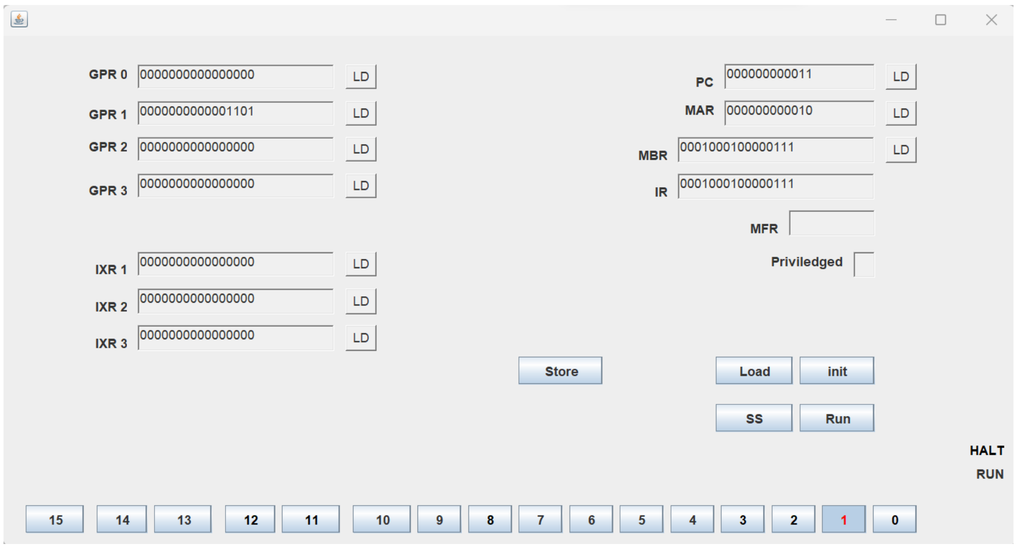
i. Load GPR0 with B: 0111
ii. Load MBR with B: 0011110000001111
iii. Load MAR with B: 0011
iv. Set PC to B: 0011
v. Click store
vi. Click SS
vii. PC is updated to B: 1111


GPR 0 0000000000000111
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 0000000000001111
MAR 000000000011
MBR 0011110000001111
IR 0011110000001111
MFR
Priviledged

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

Opcode: 04

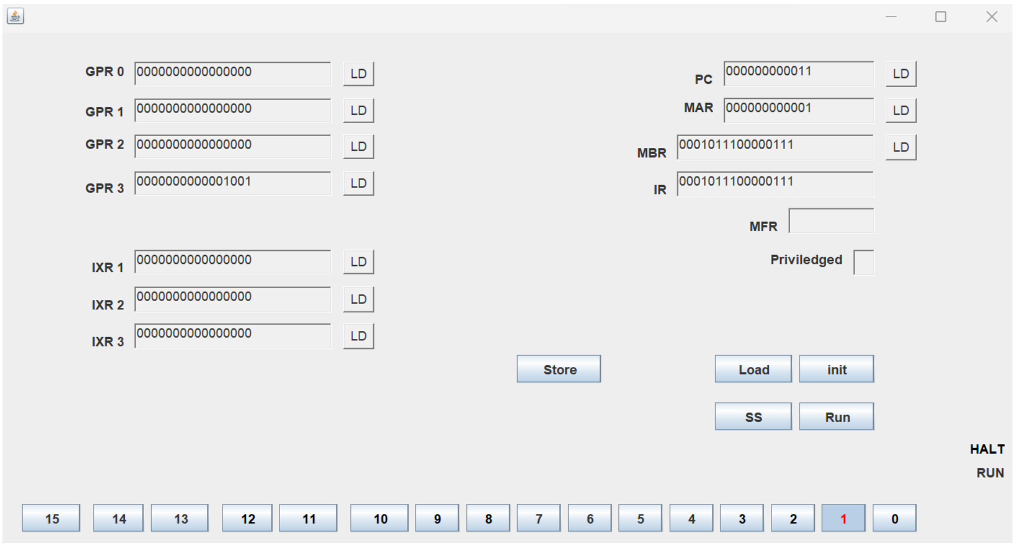
i. Load GPR1 with B: 0111
ii. Load MAR with B: 111 and MBR with B: 0110, store
iii. Load MAR with B: 0010 and MBR with B: 0001000100000111
iv. Set PC to B: 0010
v. Click on store
vi. Click SS
vii. GPR1 becomes c(r) +c(EA)= B: 0110+ B: 0111 = B: 1101


GPR 0 0000000000000000
GPR 1 0000000000001101
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000000011
MAR 000000000010
MBR 0001000100000111
IR 0001000100000111
MFR
Priviledged

14. SMR r, x, address[,I] Subtract Memory From Register, r = 0..3 r<– c(r) – c(EA)

Opcode: 05

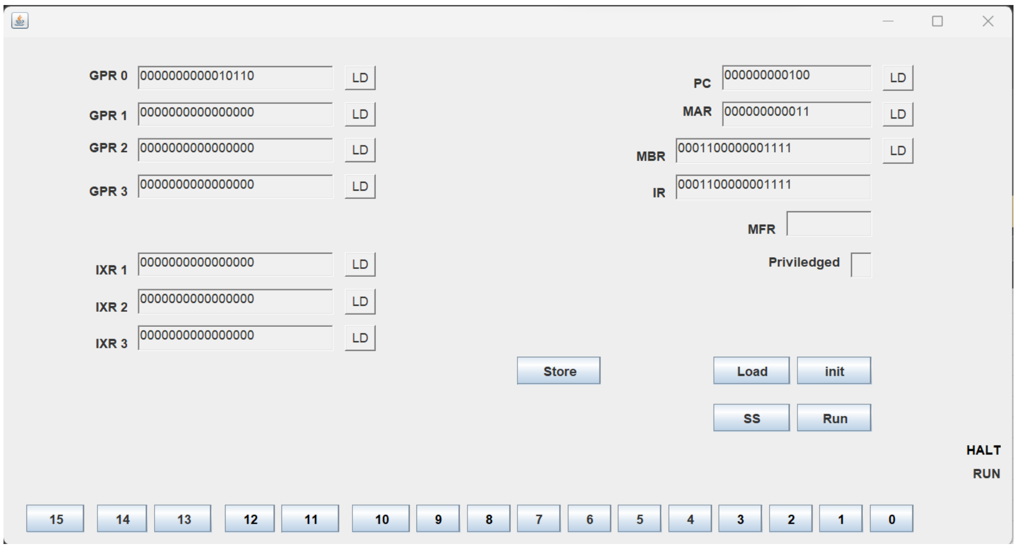
i. Load GPR3 with B: 1111
ii. Load MAR with B: 0111 and MBR with B: 0110, store
iii. Load MAR with B: 0010 and set MBR with B: 0001011100000111
iv. Set PC to B: 0010
v. Click on store
vi. Click SS
vii. GPR3 becomes c(r)-c(EA) = B: 1111- B: 110 = B: 1001


GPR 0 0000000000000000
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000001001
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000000011
MAR 000000000001
MBR 0001011100000111
IR 0001011100000111
MFR
Priviledged

15. AIR r, immed Add Immediate to Register, r = 0..3 r <– c(r) + 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

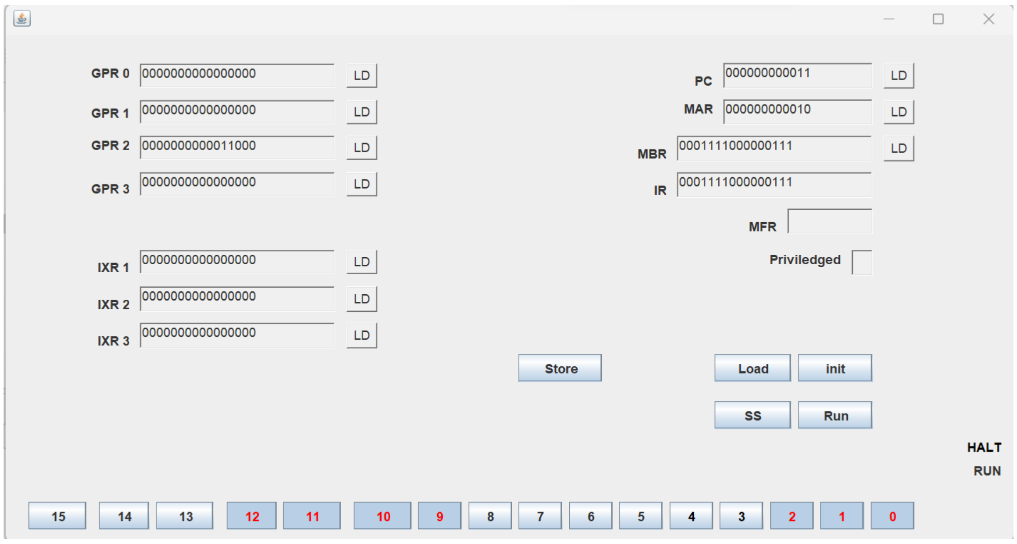
i. Load GPR0 with B: 0111
ii. Load MAR with B: 0011 and MBR with B: 0001100000001111
iii. Set PC to B: 0011
iv. Click on store
v. Click SS
vi. GPR0 becomes c(r)+immed= B: 0111+B: 1111= B: 10110


GPR 0 0000000000010110
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000000100
MAR 000000000011
MBR 0001100000001111
IR 0001100000001111
MFR
Priviledged

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

Opcode: 07

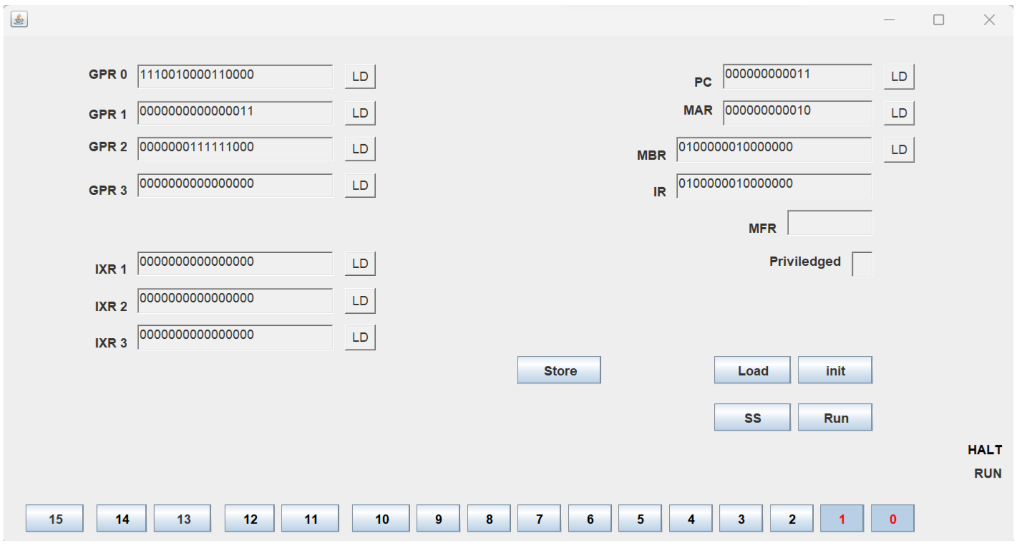
i. Load GPR2 with B: 11111
ii. Load MAR with B: 0010 and MBR with B: 0001111000000111
iii. Set PC to B: 0010
iv. Click on store
v. Click SS
vi. GPR2 becomes c(r)- immediate value = B: 11111 – B: 00111 = B: 11000


GPR 0 0000000000000000
GPR 1 0000000000000000
GPR 2 0000000000011000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000000011
MAR 000000000010
MBR 0001111000000111
IR 0001111000000111
MFR
Priviledged

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

Opcode: 20

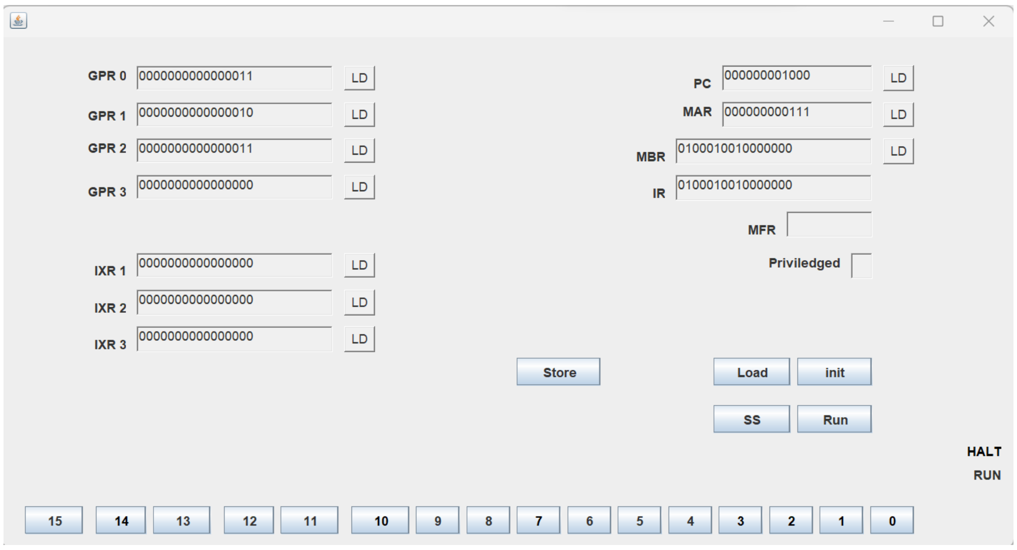
i. Load GPR0 with B: 0000000111111010
ii. Load GPR2 with B: 0000000111111000
iii. Load MAR with B: 0010 and MBR to B: 0100000010000000
iv. Set PC to B: 0010
v. Click on store
vi. Click SS
vii. Result is B: 111111010 * B: 111111000 = B: 111110010000110000
viii. First 4 bits are stored in rx which is GPR0, other 4 bits are stored at rx+1 which is GPR1


GPR 0 1110010000110000
GPR 1 0000000000000011
GPR 2 0000000111111000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000000011
MAR 000000000010
MBR 0100000010000000
IR 0100000010000000
MFR
Priviledged

18. DVD rx,ry Divide Register by Register rx, rx+1 <- 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

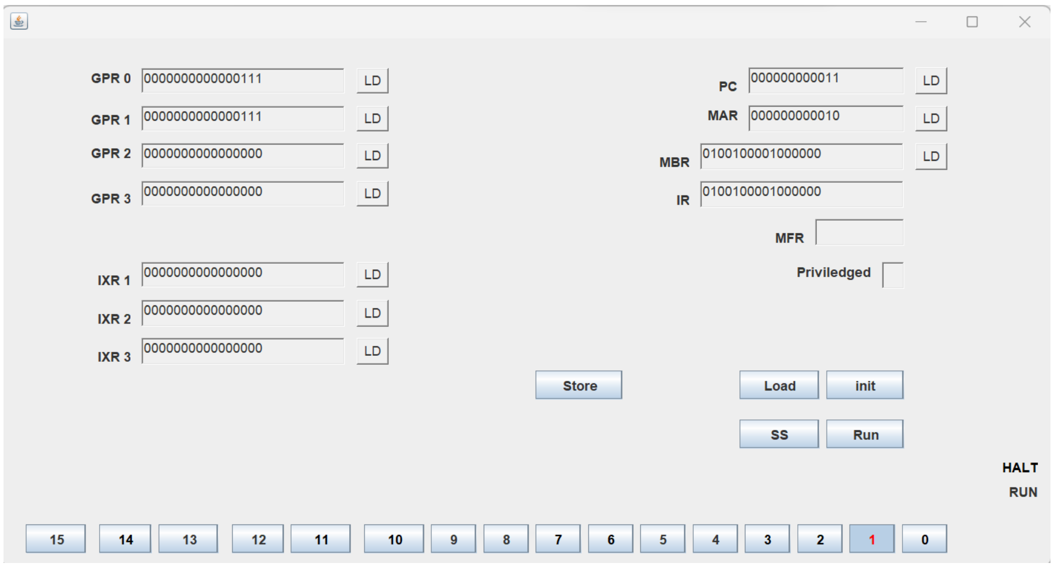


GPR 0 0000000000000011
GPR 1 0000000000000010
GPR 2 0000000000000011
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000001000
MAR 000000000111
MBR 0100010010000000
IR 0100010010000000
MFR
Priviledged

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

Opcode: 22

i. Load GPR0 with B: 0111
ii. Load GPR1 with B: 0111
iii. Load MAR with B: 0010 and MBR with B: 0100100001000000
iv. Set PC to B: 0010
v. Click store
vi. Click SS

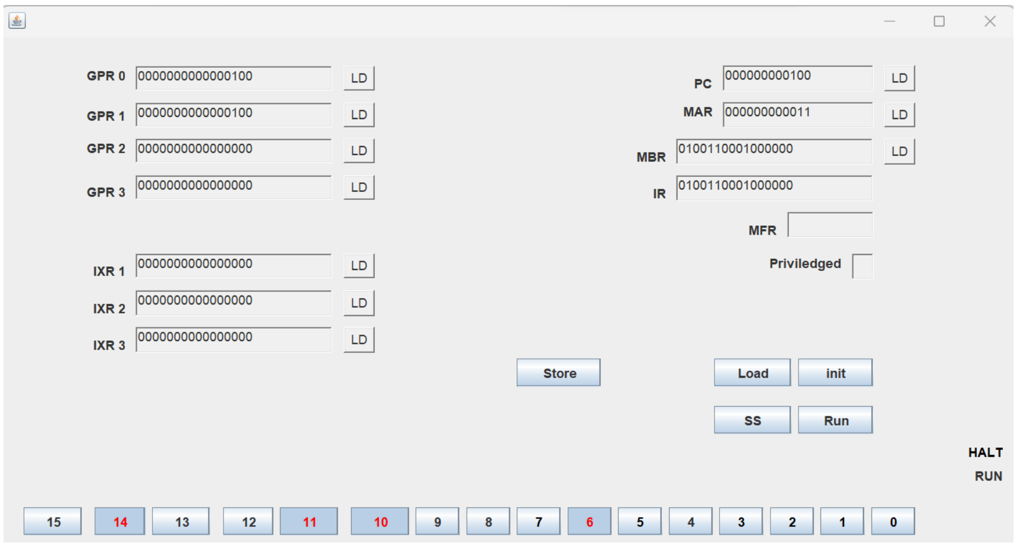


GPR 0 0000000000000111
GPR 1 0000000000000111
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000000011
MAR 000000000010
MBR 0100100001000000
IR 0100100001000000
MFR
Priviledged

20. AND rx, ry Logical And of Register and Register c(rx) <– c(rx) AND c(ry)

Opcode: 23

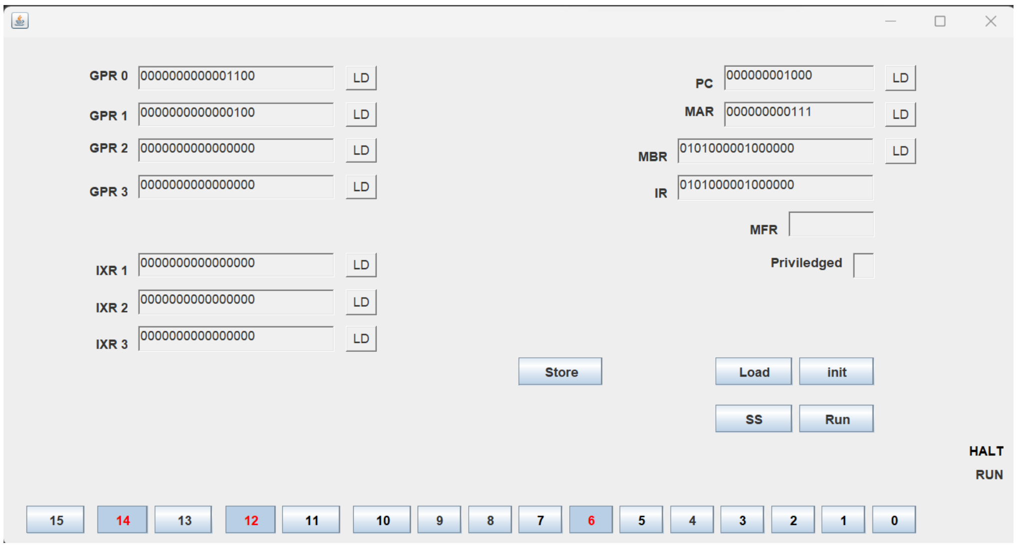
i. Set GPR0 to B: 1101
ii. Set GPR1 to B: 0100
iii. Set MAR to B: 0011 and MBR with B: 0100110001000000
iv. Set PC to B: 0011
v. Click on store
vi. Click SS
vii. GPR0 populates with B: 0100


GPR 0 0000000000000100
GPR 1 0000000000000100
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000000100
MAR 000000000011
MBR 0100110001000000
IR 0100110001000000
MFR
Priviledged

21. ORR rx, ry Logical Or of Register and Register c(rx) <– c(rx) 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

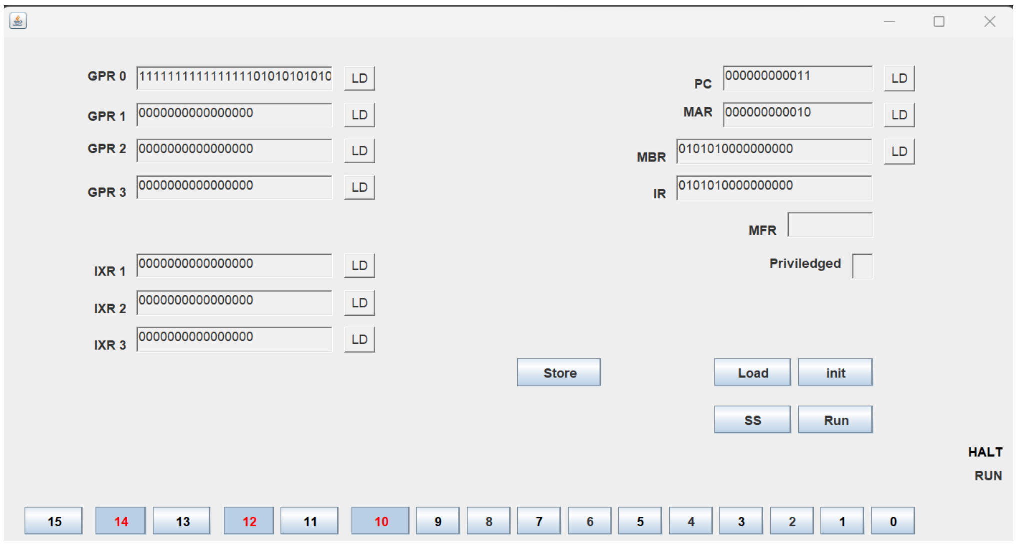


GPR 0 0000000000001100
GPR 1 0000000000000100
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000001000
MAR 000000000111
MBR 0101000001000000
IR 0101000001000000
MFR
Priviledged

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

Opcode: 25

i. Load GPR0 with B: 1010101010101010
ii. Load MAR with B: 0010 and MBR with B: 0101010000000000
iii. Set PC to B: 0010
iv. Click on store
v. Click SS
vi. GPR0 populates to B: 101010101010101

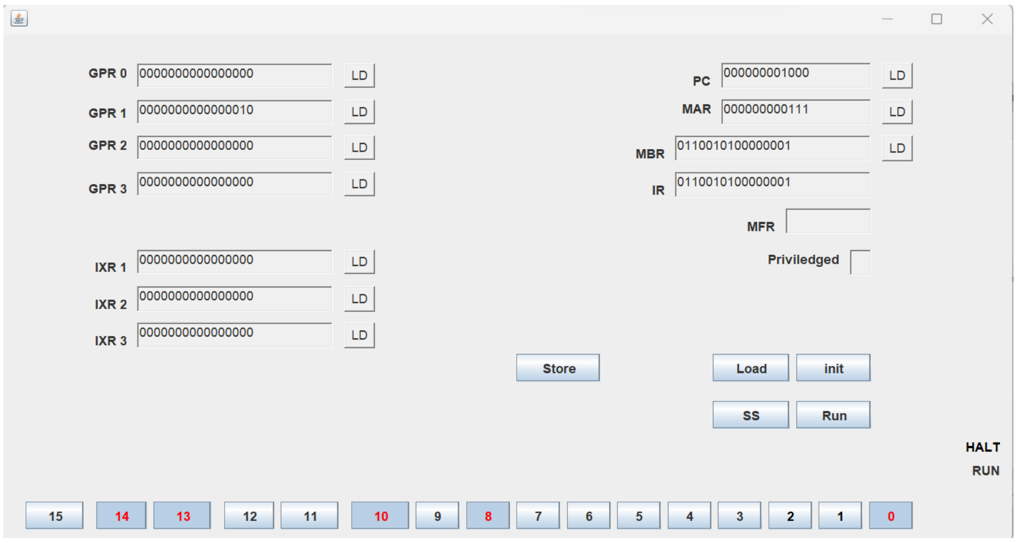


GPR 0 111111111111111101010101010
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000000011
MAR 000000000010
MBR 0101010000000000
IR 0101010000000000
MFR
Priviledged

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


GPR 0 0000000000000000
GPR 1 0000000000000010
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
PC 000000001000
MAR 000000000111
MBR 0110010100000001
IR 0110010100000001
MFR
Priviledged

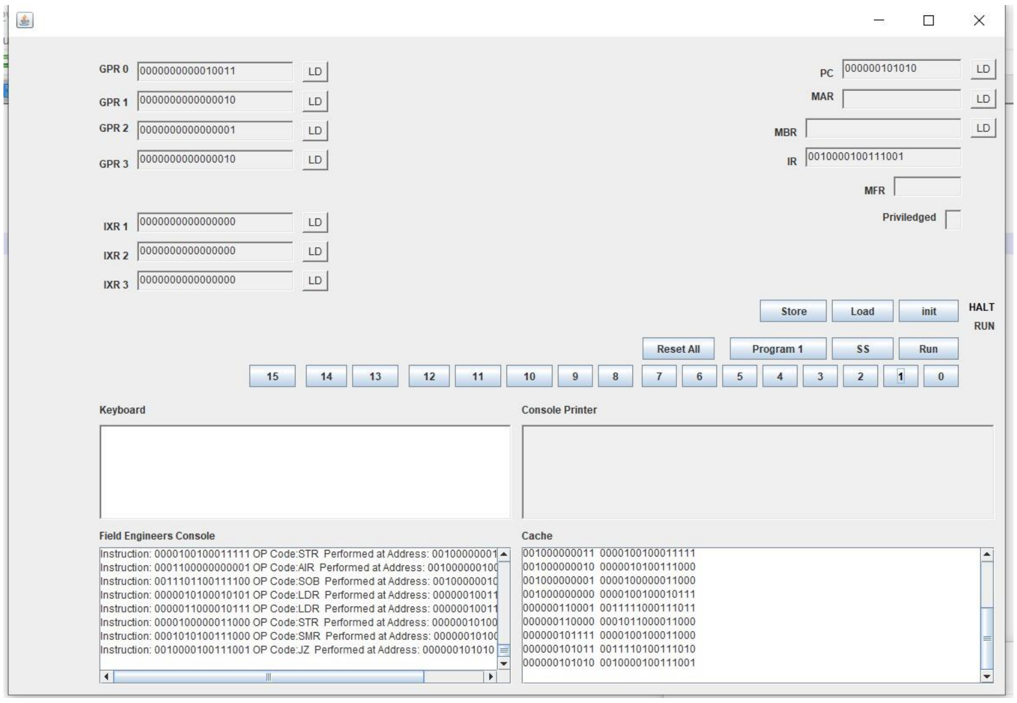
24. RRC r, count, L/R, A/L Rotate Register by Count c(r) is rotated left (L/R = 1) or right (L/R =0) either logically (A/L =1) XX, XXX is ignored Count = 0...15 If Count = 0, no rotate occurs

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.



GPR 0: 0000000000010011
GPR 1: 0000000000000010
GPR 2: 0000000000000001
GPR 3: 0000000000000010
IXR 1: 0000000000000000
IXR 2: 0000000000000000
IXR 3: 0000000000000000
PC: 000000101010
MAR:
MBR:
IR: 0010000100111001
MFR:

Field Engineers Console

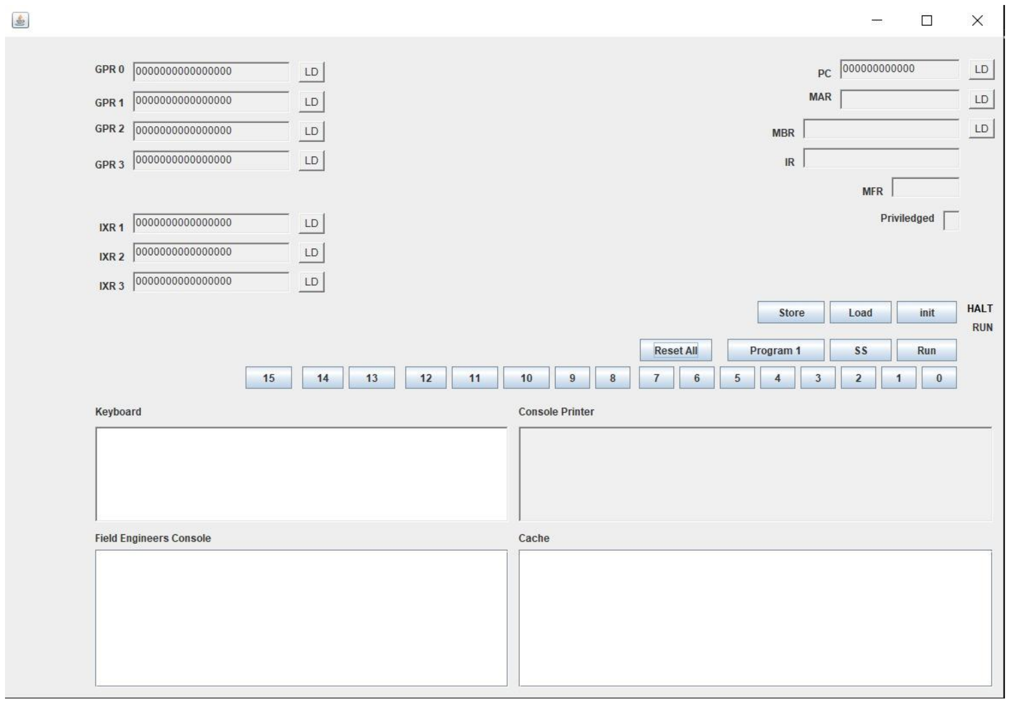
```
Instruction: 0000100100011111 OP Code:STR  Performed at Address: 00100000001
Instruction: 0001100000000001 OP Code:AIR  Performed at Address: 00100000010
Instruction: 0011101100111100 OP Code:SOB  Performed at Address: 00100000010
Instruction: 0000010100010101 OP Code:LDR  Performed at Address: 00000010011
Instruction: 0000011000010111 OP Code:LDR  Performed at Address: 00000010011
Instruction: 0000100000011000 OP Code:STR  Performed at Address: 00000010100
Instruction: 0001010100111000 OP Code:SMR  Performed at Address: 00000010100
Instruction: 0010000100111001 OP Code:JZ  Performed at Address: 000000101010
```

Cache

```
001000000011  0000100100011111
001000000010  0000010100111000
001000000001  0000100000011000
001000000000  0000100100010111
000000110001  0011111000111011
000000110000  0001011000011000
000000101111  0000100100011000
000000101011  0011110100111010
000000101010  0010000100111001
```

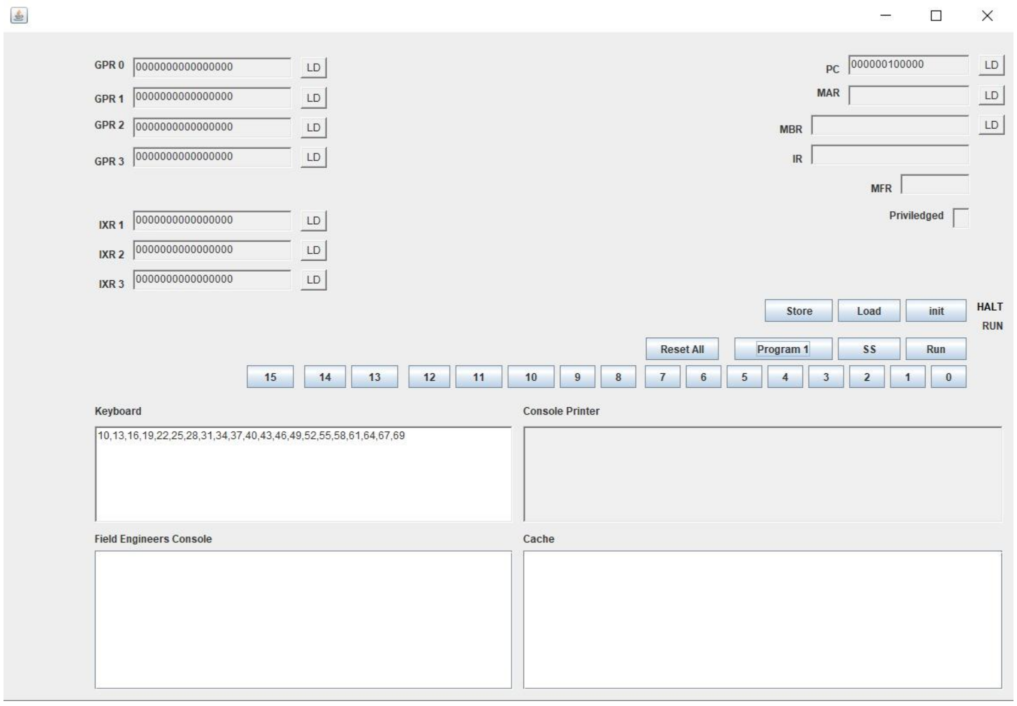
Program 1 Implementation:

1. Click on 'Reset all' button



GPR 0: 0000000000000000
GPR 1: 0000000000000000
GPR 2: 0000000000000000
GPR 3: 0000000000000000
IXR 1: 0000000000000000
IXR 2: 0000000000000000
IXR 3: 0000000000000000
PC: 000000000000
MAR:
MBR:
IR:
MFR:

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

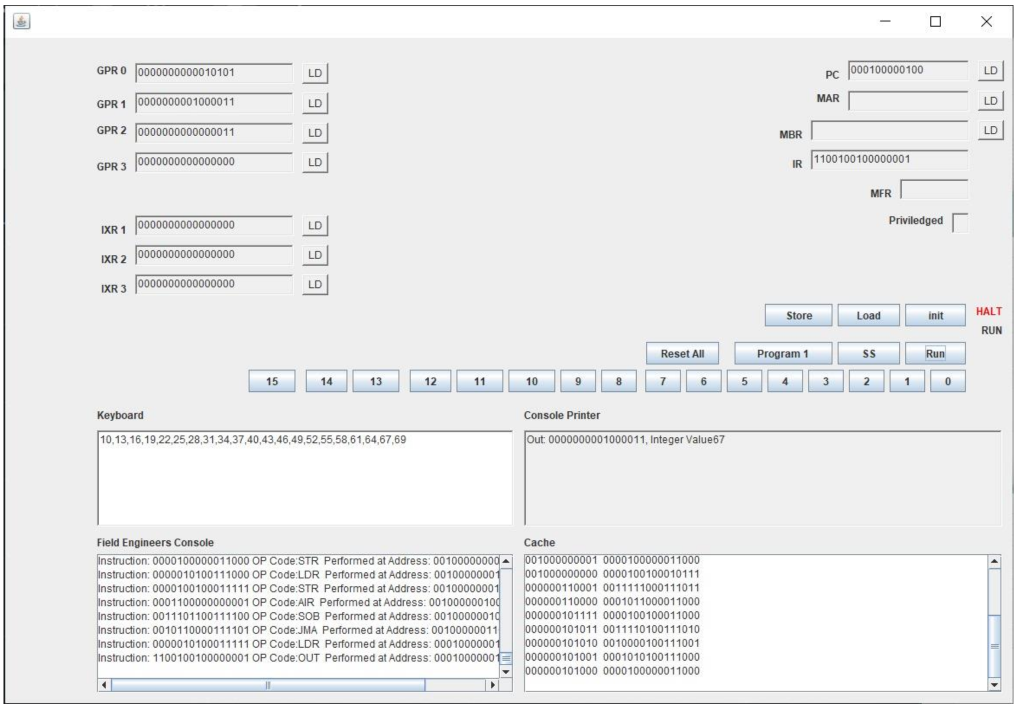


```
GPR 0  0000000000000000
GPR 1  0000000000000000
GPR 2  0000000000000000
GPR 3  0000000000000000
IXR 1  0000000000000000
IXR 2  0000000000000000
IXR 3  0000000000000000
PC     000000100000
MAR
MBR
IR
MFR
Priviledged
```

Keyboard

```
10,13,16,19,22,25,28,31,34,37,40,43,46,49,52,55,58,61,64,67,69
```

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



```
GPR 0  0000000000010101
GPR 1  0000000001000011
GPR 2  0000000000000011
GPR 3  0000000000000000
IXR 1  0000000000000000
IXR 2  0000000000000000
IXR 3  0000000000000000
PC     000100000100
MAR
MBR
IR     1100100100000001
MFR
Priviledged
```

Keyboard

```
10,13,16,19,22,25,28,31,34,37,40,43,46,49,52,55,58,61,64,67,69
```

Console Printer

```
Out: 0000000001000011, Integer Value67
```

Field Engineers Console

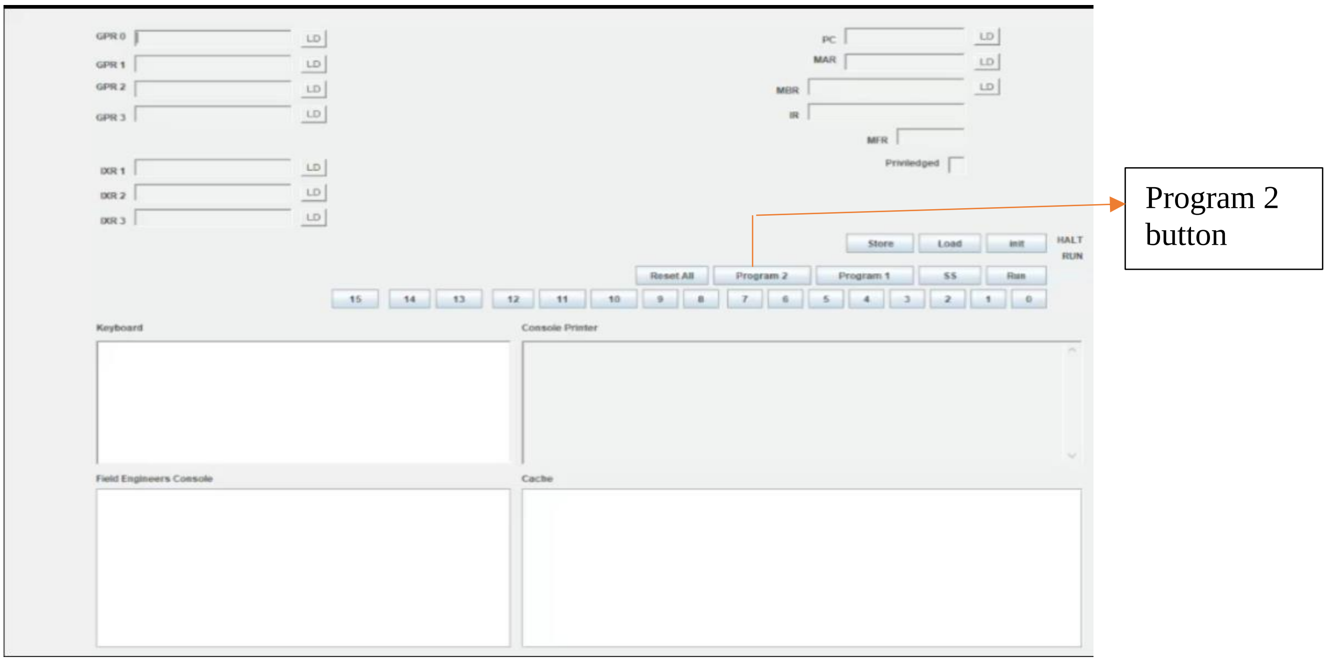
```
Instruction: 0000100000011000 OP Code:STR  Performed at Address: 00100000000
Instruction: 0000010100111000 OP Code:LDR  Performed at Address: 00100000001
Instruction: 0000100100011111 OP Code:STR  Performed at Address: 00100000001
Instruction: 0001100000000001 OP Code:AIR  Performed at Address: 00100000010
Instruction: 0011101100111100 OP Code:SOB  Performed at Address: 00100000010
Instruction: 0010110000111101 OP Code:JMA  Performed at Address: 00100000011
Instruction: 0000010100011111 OP Code:LDR  Performed at Address: 00010000001
Instruction: 1100100100000001 OP Code:OUT  Performed at Address: 00010000001
```

Cache

```
001000000001  0000100000011000
001000000000  0000100100010111
000000110001  0011111000111011
000000110000  0001011000011000
000000101111  0000100100011000
000000101011  0011110100111010
000000101010  0010000100111001
000000101001  0001010100111000
000000101000  0000100000011000
```

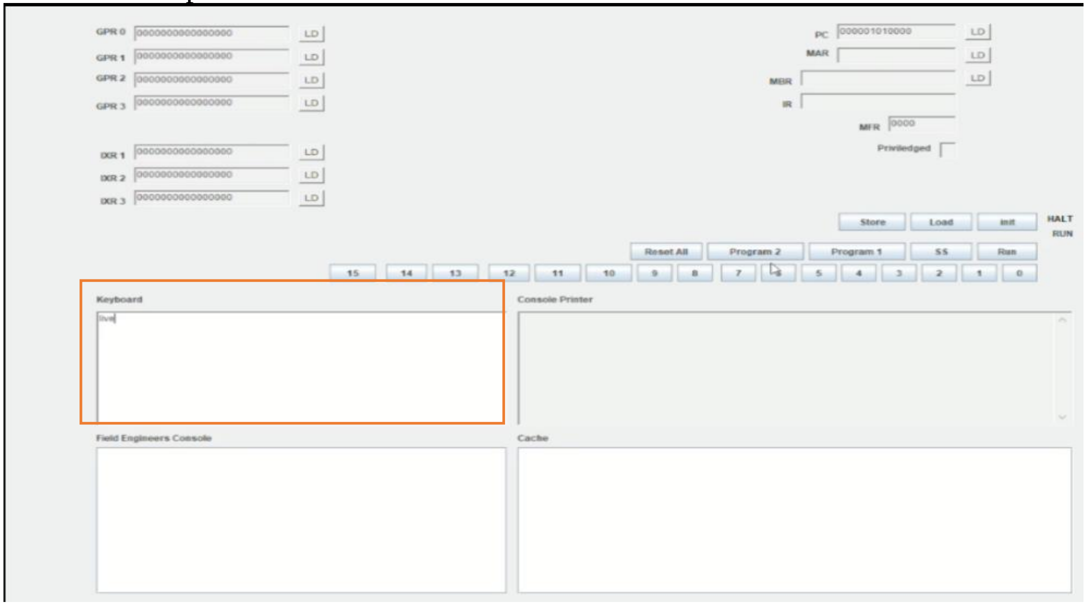
PHASE III

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.



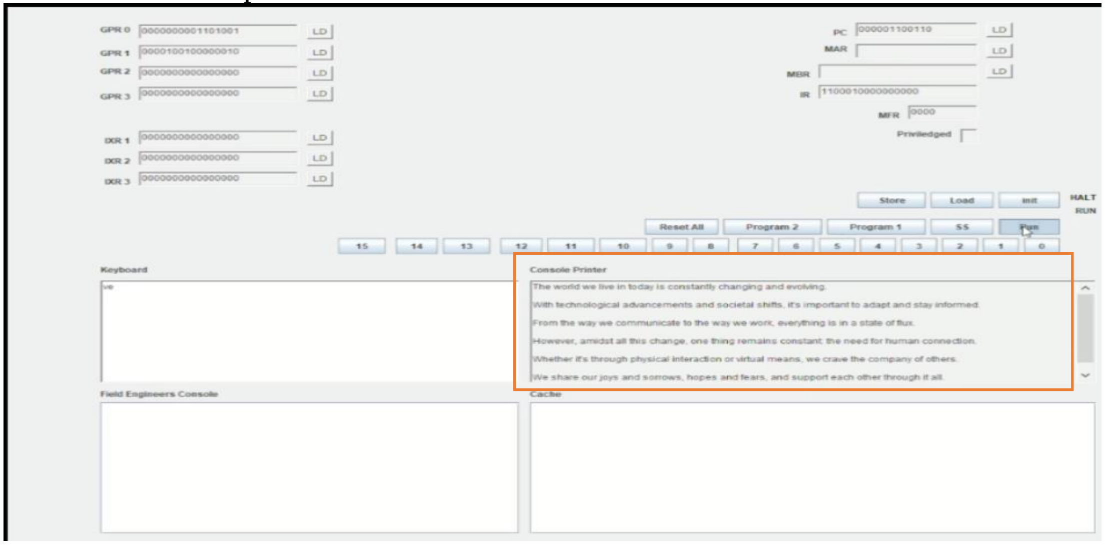
Steps to execute program 2:

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.



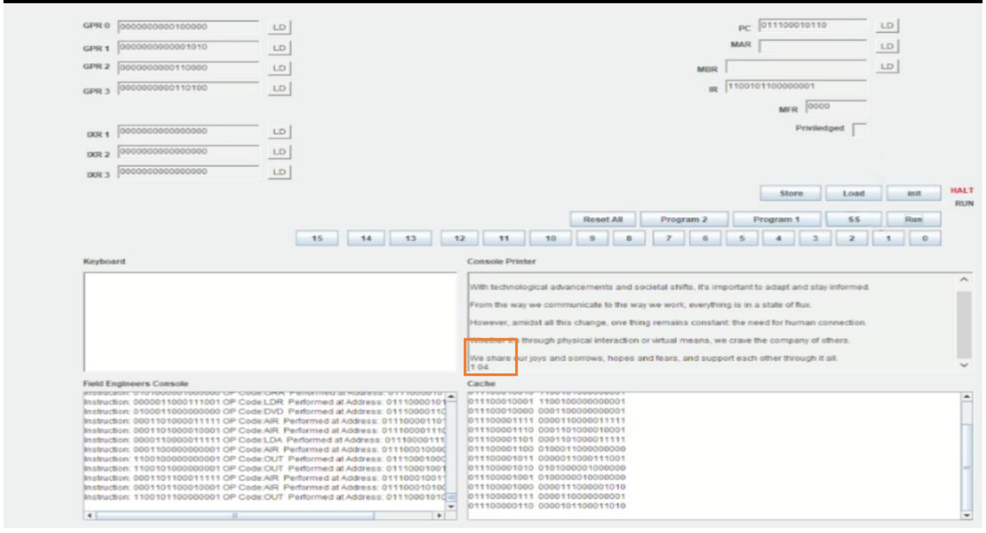
```
GPR 0  0000000000000000
GPR 1  0000000000000000
GPR 2  0000000000000000
GPR 3  0000000000000000
IXR 1  0000000000000000
IXR 2  0000000000000000
IXR 3  0000000000000000
PC     000001010000
MFR    0000
Keyboard: live
```

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.



```
GPR 0  0000000001101001
GPR 1  0000100100000010
GPR 2  0000000000000000
GPR 3  0000000000000000
IXR 1  0000000000000000
IXR 2  0000000000000000
IXR 3  0000000000000000
PC     000001100110
IR     1100010000000000
MFR    0000
Keyboard: ve
```

```
The world we live in today is constantly changing and evolving.
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.
Whether it's through physical interaction or virtual means, we crave the company of others.
We share our joys and sorrows, hopes and fears, and support each other through it all.
```



GPR 0 0000000000100000
GPR 1 0000000000001010
GPR 2 0000000000110000
GPR 3 0000000000110100

IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000

PC 011100010110
MAR
MBR
IR 1100101100000001
MFR 0000
Priviledged

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.
Whether it's through physical interaction or virtual means, we crave the company of others.
We share our joys and sorrows, hopes and fears, and support each other through it all.
1 04
```

Field Engineers Console:

```
Instruction: 0000011000111001 OP Code:LDR  Performed at Address: 0111000010
Instruction: 0100011000000000 OP Code:DVD  Performed at Address: 0111000011
Instruction: 0001101000011111 OP Code:AIR  Performed at Address: 0111000011
Instruction: 0001101000010001 OP Code:AIR  Performed at Address: 0111000011
Instruction: 0000011000011111 OP Code:LDA  Performed at Address: 0111000011
Instruction: 0001100000000001 OP Code:AIR  Performed at Address: 0111000100
Instruction: 1100100000000001 OP Code:OUT  Performed at Address: 0111000100
Instruction: 1100101000000001 OP Code:OUT  Performed at Address: 0111000100
Instruction: 0001101100011111 OP Code:AIR  Performed at Address: 0111000100
Instruction: 0001101100010001 OP Code:AIR  Performed at Address: 0111000101
Instruction: 1100101100000001 OP Code:OUT  Performed at Address: 0111000101
```

Cache:

```
011100010001  1100100000000001
011100010000  0001100000000001
011100001111  0000011000011111
011100001110  0001101000010001
011100001101  0001101000011111
011100001100  0100011000000000
011100001011  0000011000111001
011100001010  0101000001000000
011100001001  0100000010000000
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 address2 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.

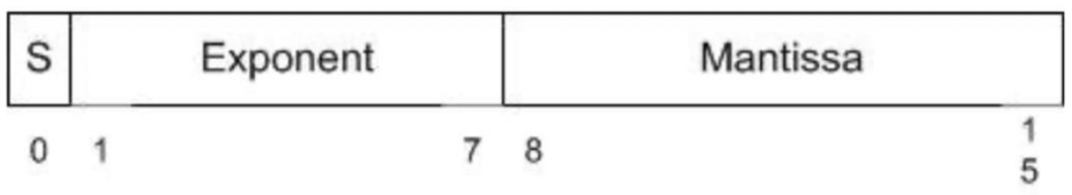
The possible machine faults that are predefined are:

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:



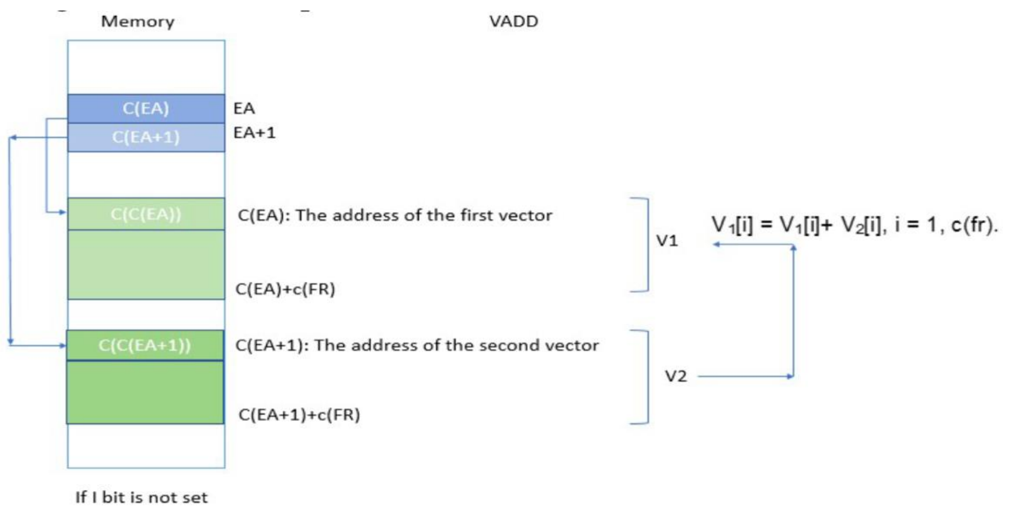
In the 16-bit binary, first bit denotes the sign of float value, second 8 bits denotes the exponent value and last 8 bits denotes the mantissa. Using sign, exponent and mantissa the actual float value can be evaluated.

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

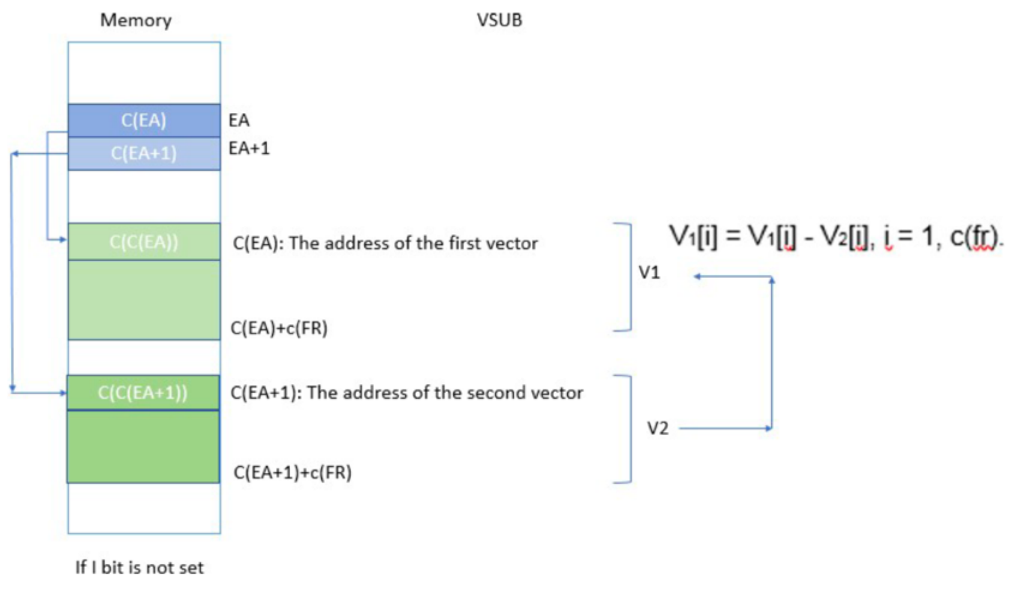
The below diagram sows the implementation of VADD in the simulator.


Memory
VADD
C(EA)
EA
C(EA+1)
EA+1
C(C(EA))
C(EA): The address of the first vector
C(EA)+c(FR)
V1
$V_1[i] = V_1[i] + V_2[i]$, i = 1, c(fr).
C(C(EA+1))
C(EA+1): The address of the second vector
V2
C(EA+1)+c(FR)
If I bit is not set

VSUB:

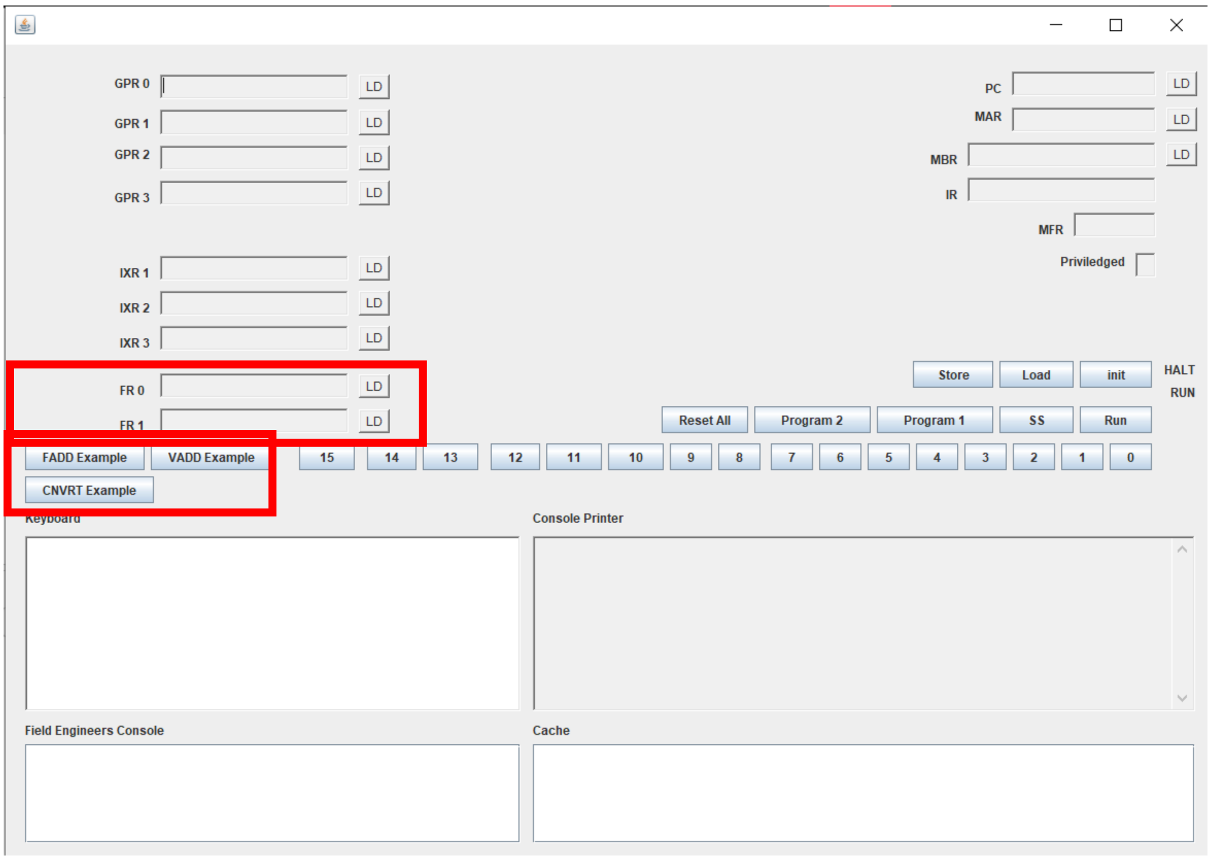
036	VSUB fr, x, address[,I]	Vector Subtract fr contains the length of the vectors c(EA) or c(c(EA)), if I bit set is address of first vector c(EA+1) or c(c(EA+1)), if I bit set is address of the second vector Let V_1 be vector at address; Let V_2 be vector at address+1 Then, $V_1[i] = V_1[i] - V_2[i]$,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

The diagram below shows the implementation of VSUB:


Memory
VSUB
C(EA)
EA
C(EA+1)
EA+1
C(C(EA))
C(EA): The address of the first vector
$V_1[i] = V_1[i] - V_2[i]$, i = 1, c(fr).
V1
C(EA)+c(FR)
C(C(EA+1))
C(EA+1): The address of the second vector
V2
C(EA+1)+c(FR)
If I bit is not set

Updated UI elements:

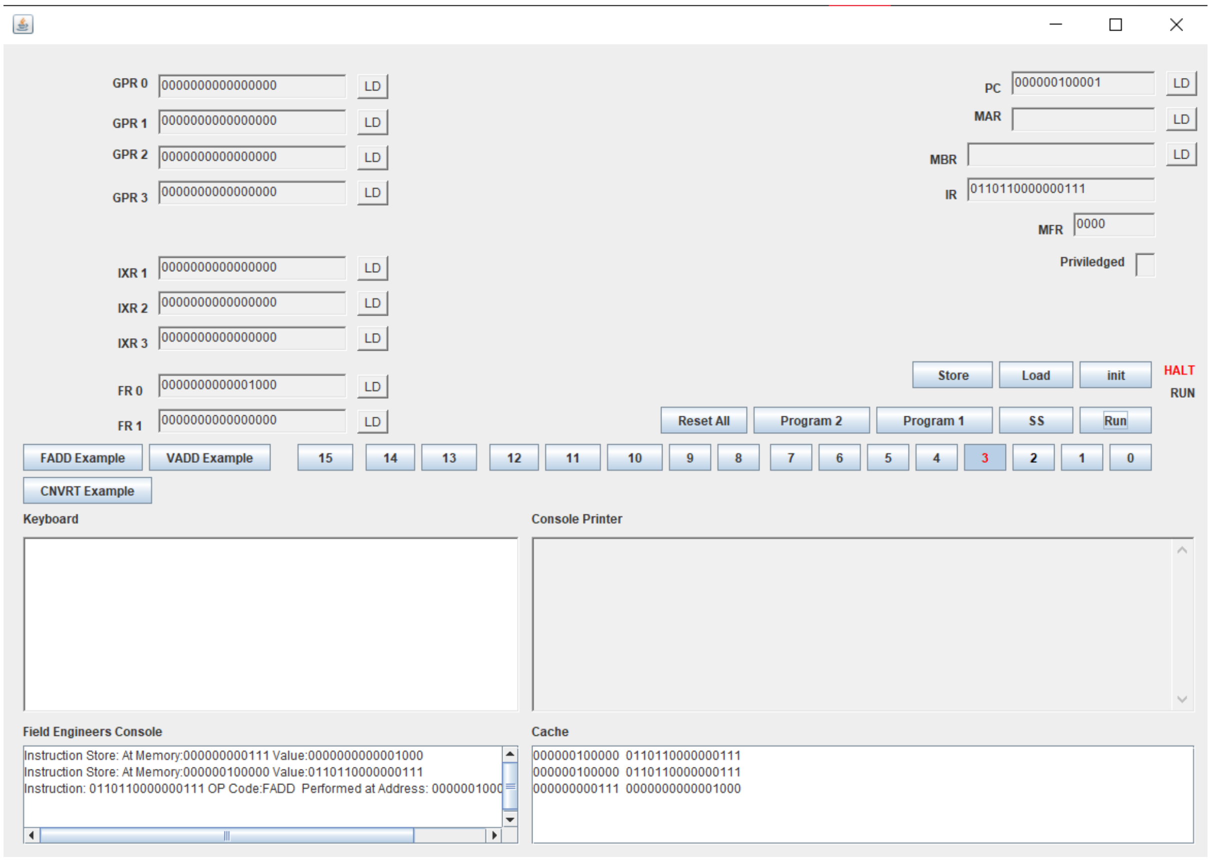
FR0, FR1: Floating point register, for floating point operations.



Execution Steps:

1. FADD fr, x, address[,I] Floating Add Memory To Register c(fr) <- c(fr) + c(EA), c(fr) <- c(fr) + c(c(EA)), if I bit set fr must be 0 or 1. OVERFLOW may be set
Opcode: 33

i. Set MAR to B:111 and MBR to B: 0000000000001000.
ii. Set MAR to B: 1000 and MBR to B: 0110110000000111.
iii. Press store to store MBR value at MAR address.
iv. Set PC to B: 1000
v. Click SS
vi. The FR0 gets updated with the result.



GPR 0	0000000000000000
GPR 1	0000000000000000
GPR 2	0000000000000000
GPR 3	0000000000000000
IXR 1	0000000000000000
IXR 2	0000000000000000
IXR 3	0000000000000000
FR 0	0000000000001000
FR 1	0000000000000000
PC	000000100001
MAR	
MBR	
IR	0110110000000111
MFR	0000
Priviledged	

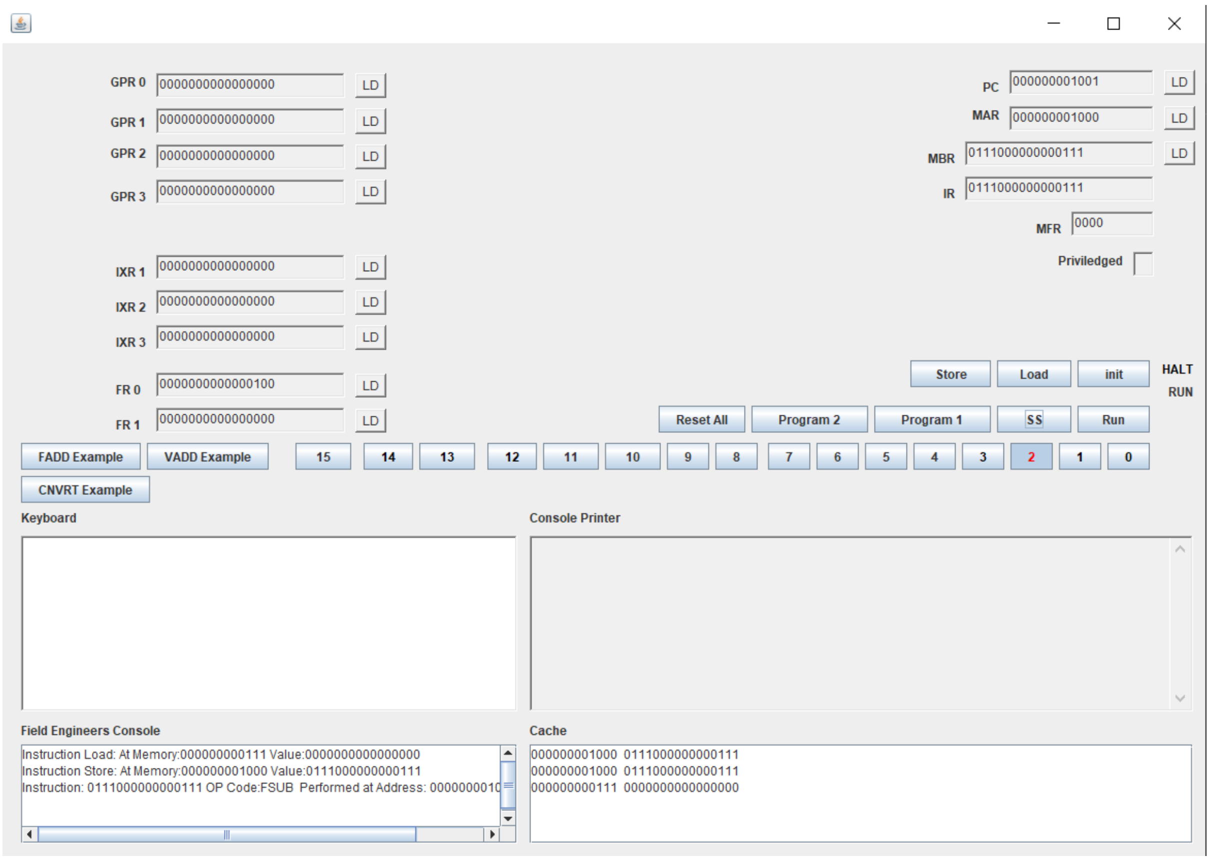
Field Engineers Console

```
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) <- c(fr) - c(EA), c(fr) <- 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



GPR 0 0000000000000000
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
FR 0 0000000000000100
FR 1 0000000000000000

PC 000000001001
MAR 000000001000
MBR 0111000000000111
IR 0111000000000111
MFR 0000

Field Engineers Console

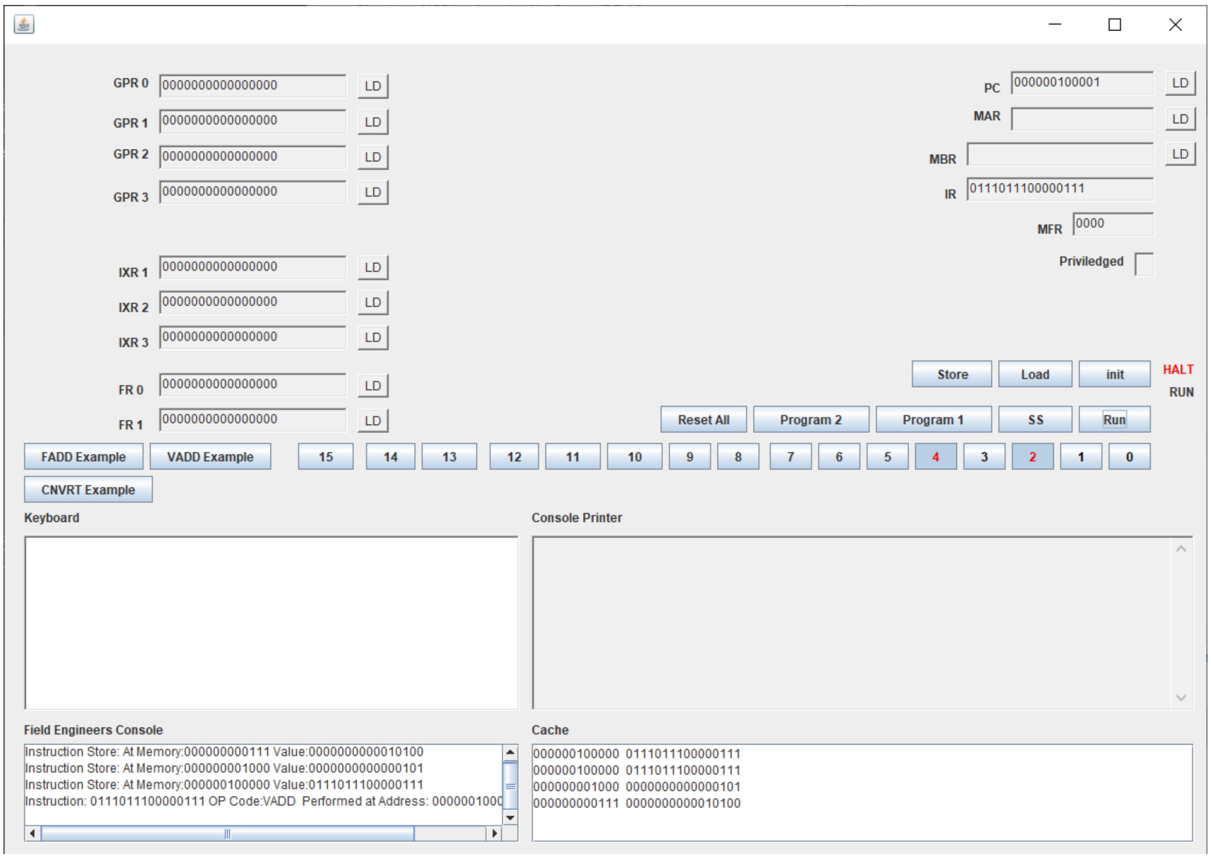
```
Instruction Load: At Memory:000000000111 Value:0000000000000000
Instruction Store: At Memory:000000001000 Value:0111000000000111
Instruction: 0111000000000111 OP Code:FSUB  Performed at Address: 00000000010
```

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:0000000000010100.
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.



GPR 0	0000000000000000
GPR 1	0000000000000000
GPR 2	0000000000000000
GPR 3	0000000000000000
IXR 1	0000000000000000
IXR 2	0000000000000000
IXR 3	0000000000000000
FR 0	0000000000000000
FR 1	0000000000000000
PC	000000100001
MAR	
MBR	
IR	0111011100000111
MFR	0000
Priviledged	

Field Engineers Console

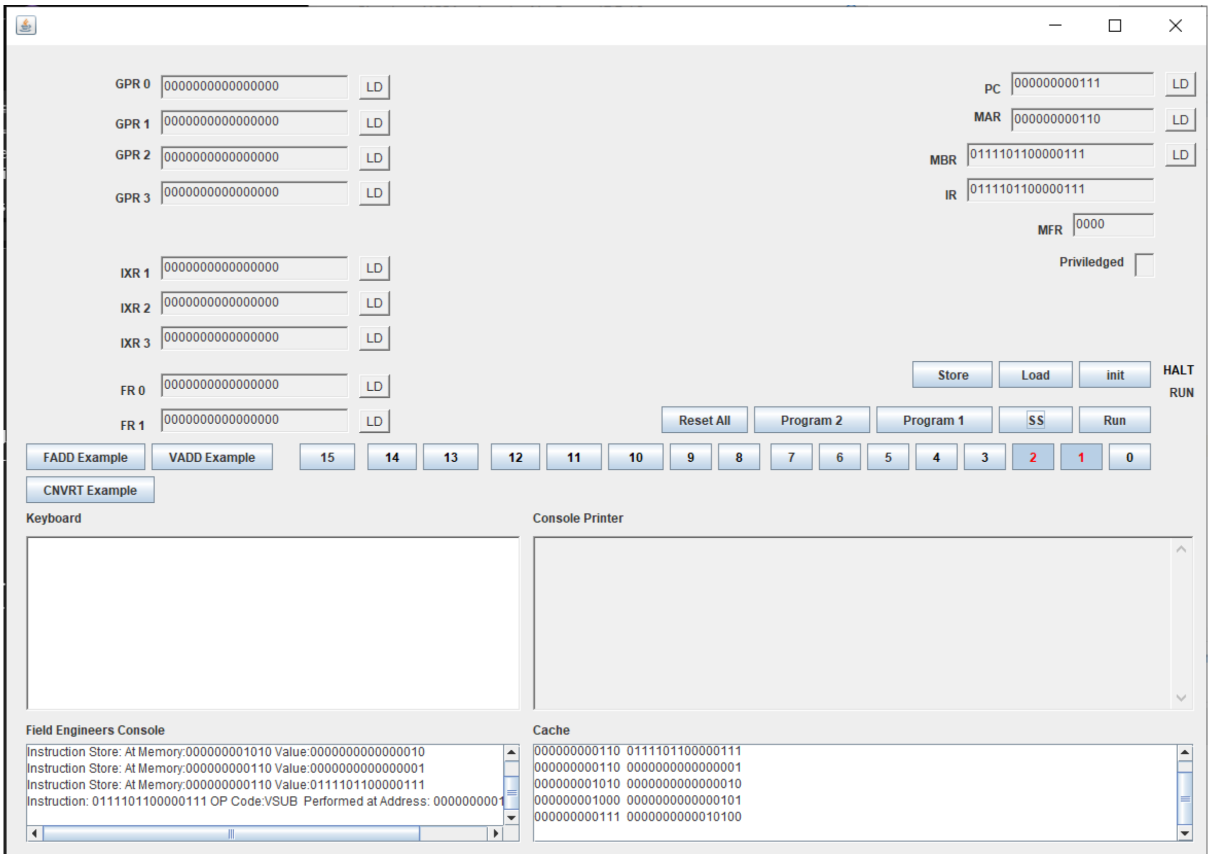
```
Instruction Store: At Memory:000000000111 Value:0000000000010100
Instruction Store: At Memory:000000001000 Value:0000000000000101
Instruction Store: At Memory:000000100000 Value:0111011100000111
Instruction: 0111011100000111 OP Code:VADD  Performed at Address: 0000001000
```

Cache

```
000000100000  0111011100000111
000000100000  0111011100000111
000000001000  0000000000000101
000000000111  0000000000010100
```

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:0000000000010100.
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.



GPR 0	0000000000000000
GPR 1	0000000000000000
GPR 2	0000000000000000
GPR 3	0000000000000000
IXR 1	0000000000000000
IXR 2	0000000000000000
IXR 3	0000000000000000
FR 0	0000000000000000
FR 1	0000000000000000
PC	000000000111
MAR	000000000110
MBR	0111101100000111
IR	0111101100000111
MFR	0000

Field Engineers Console

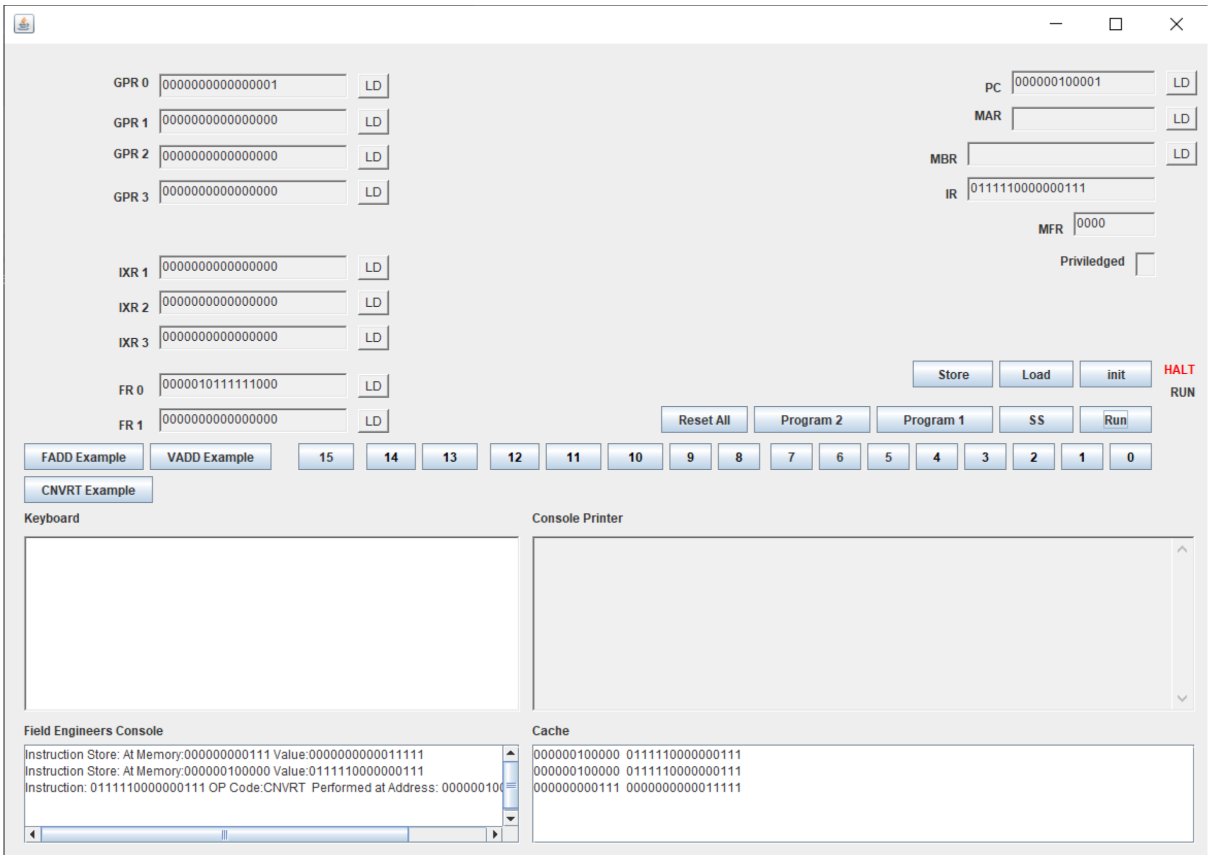
```
Instruction Store: At Memory:000000001010 Value:0000000000000010
Instruction Store: At Memory:000000000110 Value:0000000000000001
Instruction Store: At Memory:000000000110 Value:0111101100000111
Instruction: 0111101100000111 OP Code:VSUB  Performed at Address: 0000000001
```

Cache

```
000000000110  0111101100000111
000000000110  0000000000000001
000000001010  0000000000000010
000000001000  0000000000000101
000000000111  0000000000010100
```

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. 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.



GPR 0	0000000000000001	PC	000000100001
GPR 1	0000000000000000	MAR	
GPR 2	0000000000000000	MBR	
GPR 3	0000000000000000	IR	0111110000000111
IXR 1	0000000000000000	MFR	0000
IXR 2	0000000000000000	Priviledged	
IXR 3	0000000000000000		
FR 0	0000010111111000		
FR 1	0000000000000000		

HALT

Field Engineers Console

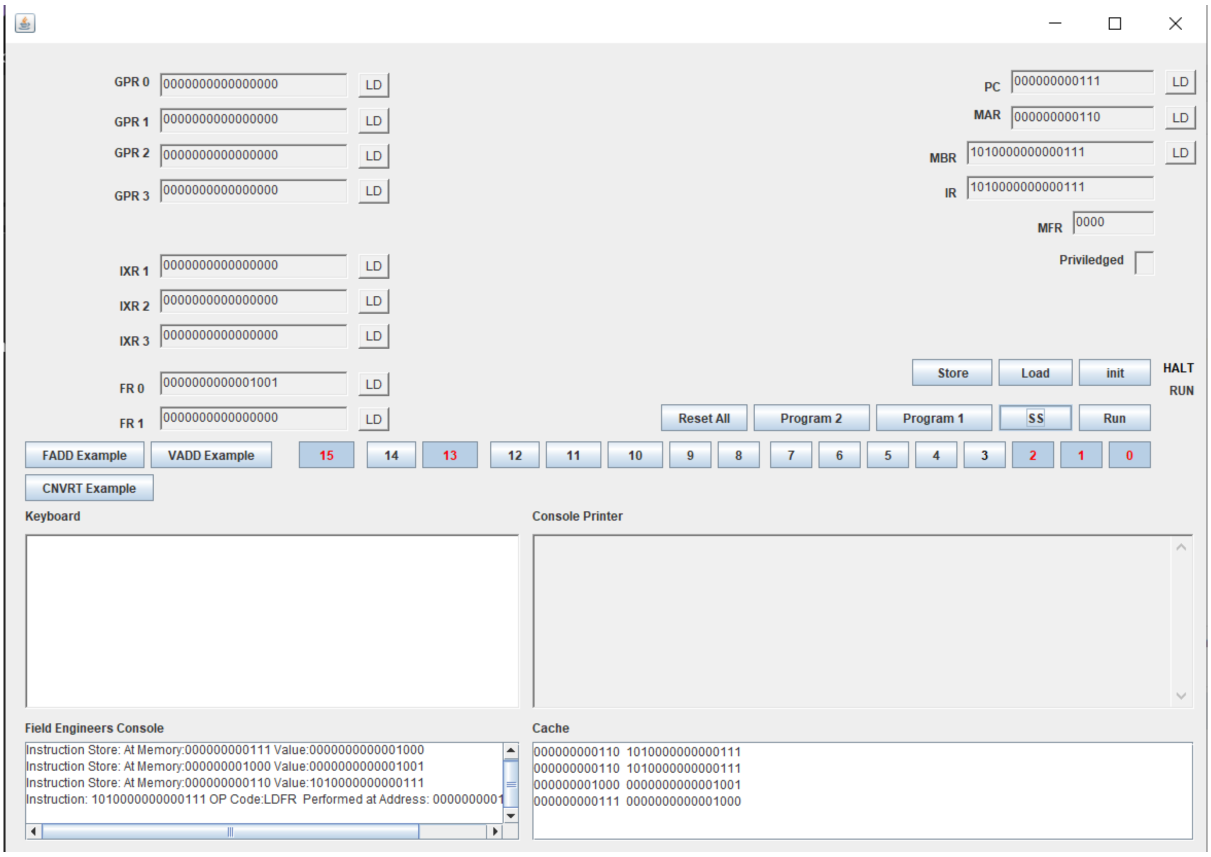
```
Instruction Store: At Memory:000000000111 Value:0000000000011111
Instruction Store: At Memory:000000100000 Value:0111110000000111
Instruction: 0111110000000111 OP Code:CNVRT  Performed at Address: 00000010(
```

Cache

```
000000100000  0111110000000111
000000100000  0111110000000111
000000000111  0000000000011111
```

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

i. Set MAR to B:111 and MBR to B: 0000000000001000).
ii. Set MAR to B: 1000 and MBR to B: 0000000000001001.
iii. Set MAR to B: 110 and MBR to B: 1010000000000111.
iv. Press store to store MBR value at MAR address.
v. Set PC to B: 110
vi. Click SS
vii. The FR0 gets updated with the value.



GPR 0 0000000000000000
GPR 1 0000000000000000
GPR 2 0000000000000000
GPR 3 0000000000000000
IXR 1 0000000000000000
IXR 2 0000000000000000
IXR 3 0000000000000000
FR 0 0000000000001001
FR 1 0000000000000000
PC 000000000111
MAR 000000000110
MBR 1010000000000111
IR 1010000000000111
MFR 0000
Priviledged

Field Engineers Console

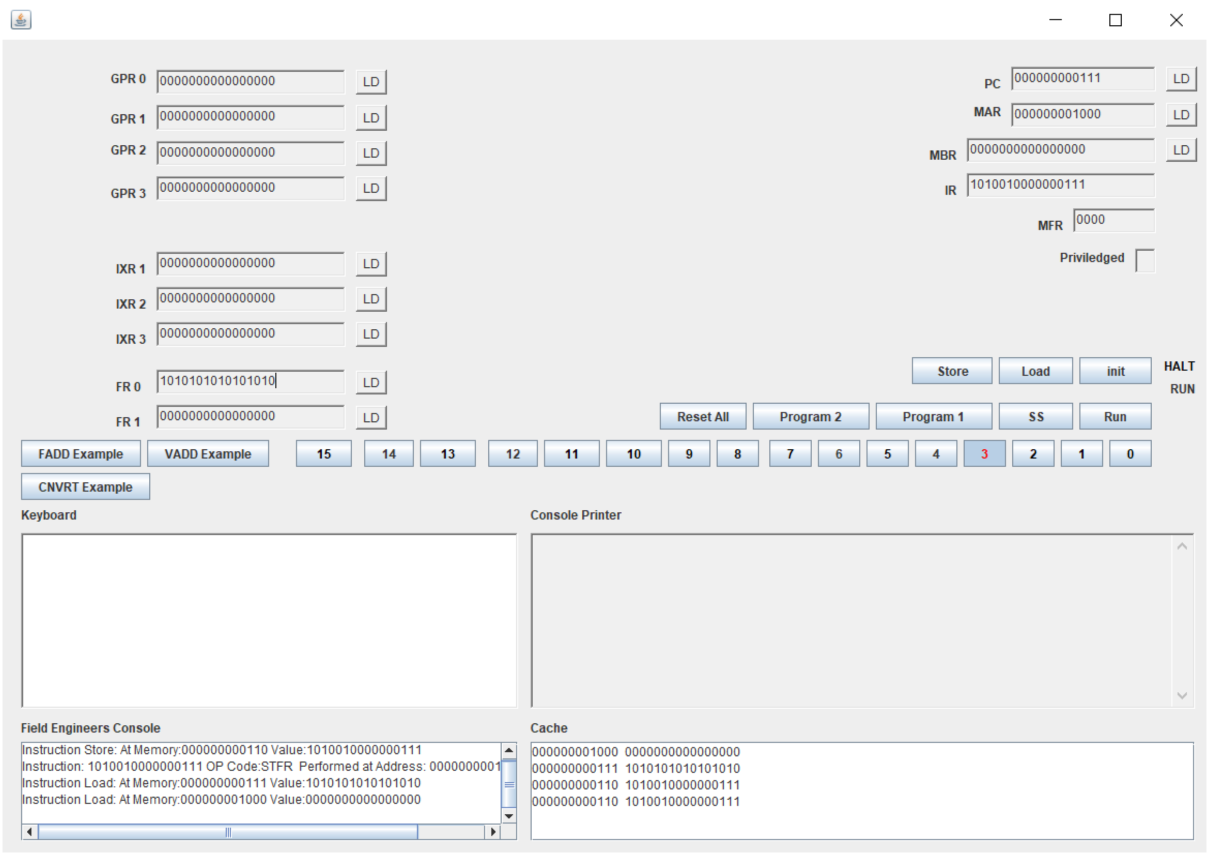
```
Instruction Store: At Memory:000000000111 Value:0000000000001000
Instruction Store: At Memory:000000001000 Value:0000000000001001
Instruction Store: At Memory:000000000110 Value:1010000000000111
Instruction: 1010000000000111 OP Code:LDFR  Performed at Address: 0000000001
```

Cache

```
000000000110  1010000000000111
000000000110  1010000000000111
000000001000  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.



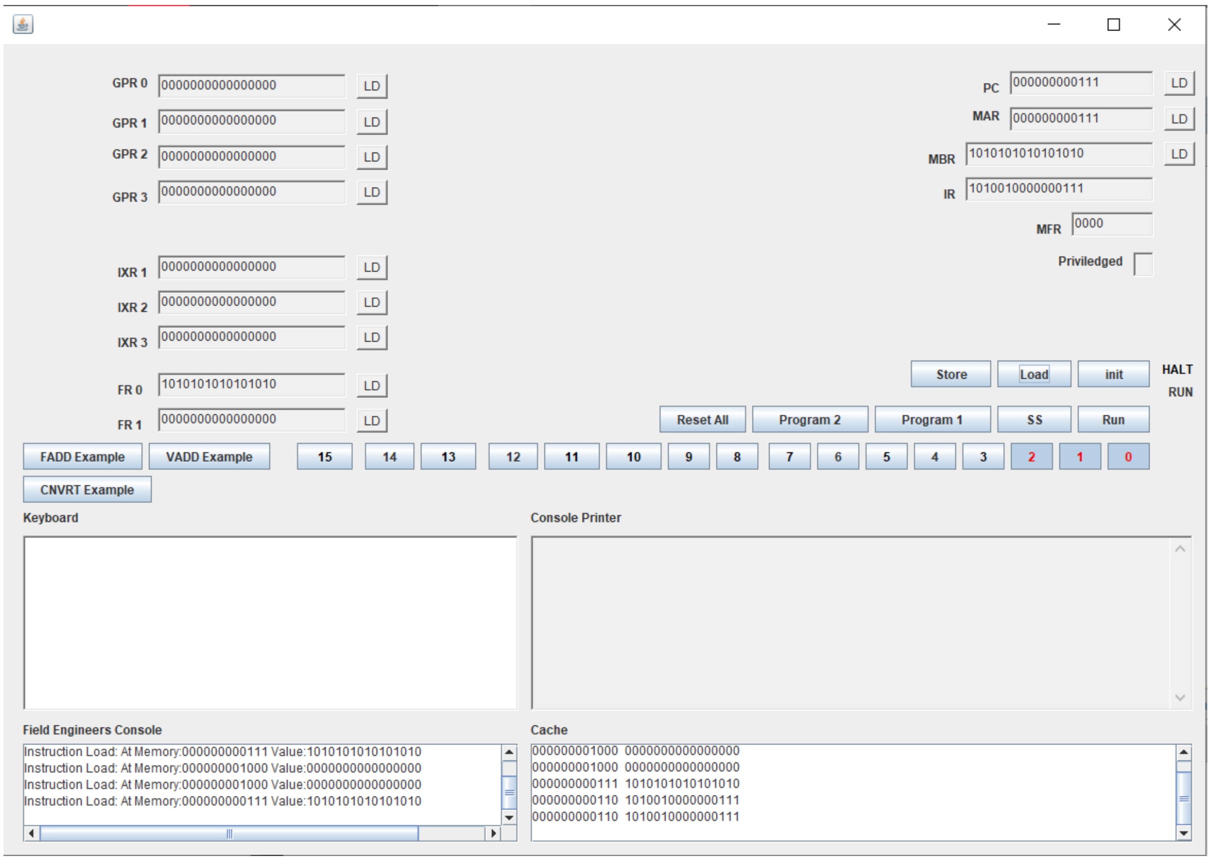
GPR 0	0000000000000000	PC	000000000111
GPR 1	0000000000000000	MAR	000000001000
GPR 2	0000000000000000	MBR	0000000000000000
GPR 3	0000000000000000	IR	1010010000000111
IXR 1	0000000000000000	MFR	0000
IXR 2	0000000000000000	Priviledged	
IXR 3	0000000000000000		
FR 0	1010101010101010		
FR 1	0000000000000000		

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
```



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

PC: 000000000111
MAR: 000000000111
MBR: 1010101010101010
IR: 1010010000000111
MFR: 0000
Priviledged: ☐

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.