Instructions for phase 3 Demo

Jeremy Case, Xinhua Fan, Teodor Georgiev, Julie Yu

Part 1 - Test Program 2

Deliverable description:
Demonstration that Program 2 works.

Demo Steps:

1. Click on "Test Program 2"

2. (**Please use only single-space and single periods**)Randomly enter **6** sentences with **period(.)** separated, such as - "Demonstration that Program 2 works. and how to operate it. Include source code for program 2. what the console layout is and how to operate it. Simple documentation describing how to use your simulator. Load instructions from a file." - Click on "ENTER"

3. Enter the **word** what you intend to search for - such as "**instructions**" - then click on "ENTER"

[Expected Result] - Output the first location and sentence number of the searched word in the six sentences. (our program will only find the first one if there are multiple matches to be found. And please reopen the GUI and get it reloaded if the result is not expected)

Part 2 - Demo for Trap/Machine Fault

Deliverable description:
Machine Fault (Part III)

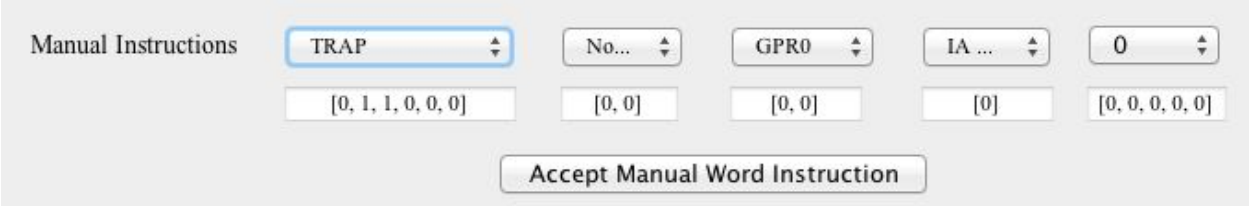
Demo Steps (Please reopen the GUI if the result is not expected)**:**

1. TRAP code

   s1. input 100 to PC

   s2. input 10 (between 0 - 15) to GPR0 as the code

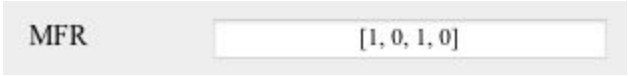
s3. select relevant values for TRAP (IX = 0; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction



[Expected Result] ->

1. PC = 100 (actually it has changed from 100[current pc] to 20[the address of the user-specified instructions stored elsewhere in memory for trap] to 100[return previous pc after executing routine])

2. MFR = 10 (contains the ID code [trap code] if a machine fault after it occurs)



3. MSR = 110 (the routine instruction is defined as 100 + trap code to indicate the status of the health of the machine)

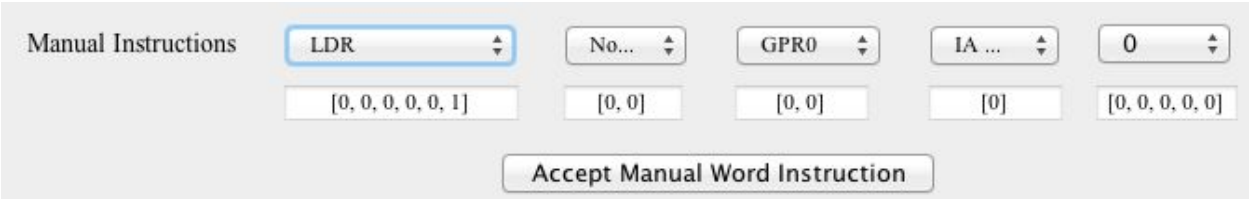
4. Memory location 0 = 20 (contains the address of the specific executed routine) - Click “Accept” close to Memory [0] to get content (20) visible

2. Machine Fault

2.1. Illegal Memory Address to Reserved Locations

s1. input 100 to PC

s2. select relevant values for LDR (IX = 0; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction



[Expected Result] ->

1. PC = 100 (actually it has changed from 100[current pc] to 11[the address of the user-specified instructions stored elsewhere in memory for trap] to 100[return previous pc after executing routine])

2. MFR = 1 (contains the ID code [id for Illegal Memory Address to Reserved Locations] if a machine fault after it occurs)

3. MSR = 101 (the routine instruction is defined as 100 + ID code to indicate the status of the health of the machine)

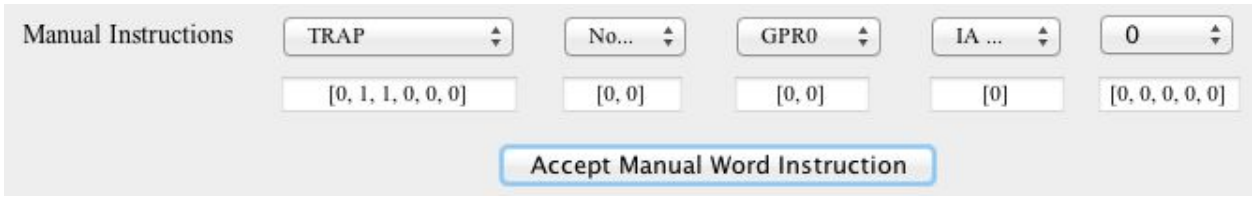
4. Memory location 1 = 11 (contains the address of the specific executed routine) - Click “Accept” close to Memory [1] to get content (11) visible

2.2. Illegal TRAP code

s1. input 100 to PC

s2. input 20 (out of 0 - 15) to GPR0 as the code

s3. select relevant values for TRAP (IX = 0; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction



[Expected Result] ->

1. PC = 100 (actually it has changed from 100[current pc] to 12[the address of the user-specified instructions stored elsewhere in memory for trap] to 100[return previous pc after executing routine])

2. MFR = 2 (contains the ID code [id for Illegal TRAP code] if a machine fault after it occurs)

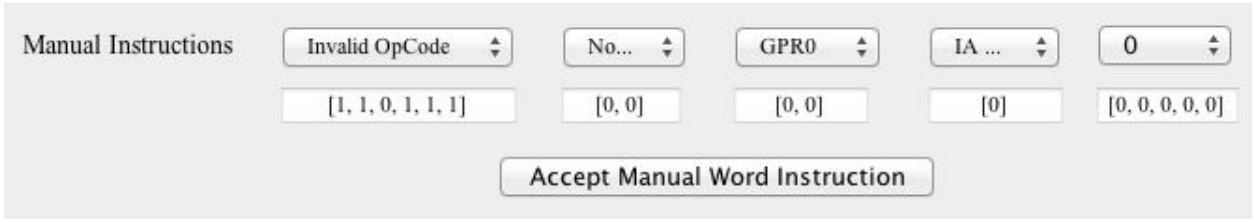
3. MSR = 102 (the routine instruction is defined as 100 + ID code to indicate the status of the health of the machine)

4. Memory location 1 = 12 (contains the address of the specific executed routine) - Click “Accept” close to Memory [1] to get content (12) visible

2.3. Illegal Operation Code

s1. input 100 to PC

s2. select relevant values for Invalid OpCode (IX = 0; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction



[Expected Result] ->

1. PC = 100 (actually it has changed from 100[current pc] to 13[the address of the user-specified instructions stored elsewhere in memory for trap] to 100[return previous pc after executing routine])

2. MFR = 3 (contains the ID code [id for Illegal Operation Code] if a machine fault after it occurs)

3. MSR = 103 (the routine instruction is defined as 100 + ID code to indicate the status of the health of the machine)

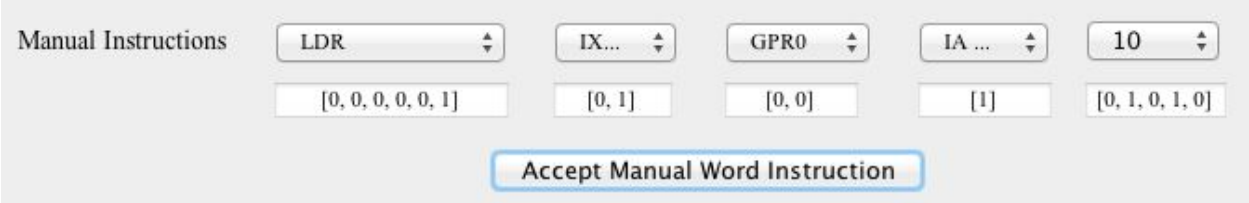
4. Memory location 1 = 13 (contains the address of the specific executed routine) - Click “Accept” close to Memory [1] to get content (13) visible

2.4. Illegal Memory Address beyond 2048 (memory installed)

s1. input 100 to PC

s2. input 2300 to IX1

s3. select relevant values for LDR (IX = 1; GPR = 0; I = 1; Address = 10) in manual input part -> accept manual word instruction



[Expected Result] ->

1. PC = 100 (actually it has changed from 100[current pc] to 14[the address of the user-specified instructions stored elsewhere in memory for trap] to 100[return previous pc after executing routine])

2. MFR = 4 (contains the ID code [id for Illegal Memory Address beyond 2048] if a machine fault after it occurs)

3. MSR = 104 (the routine instruction is defined as 100 + ID code to indicate the status of the health of the machine)

4. Memory location 1 = 14 (contains the address of the specific executed routine) - Click “Accept” close to Memory [1] to get content (14) visible

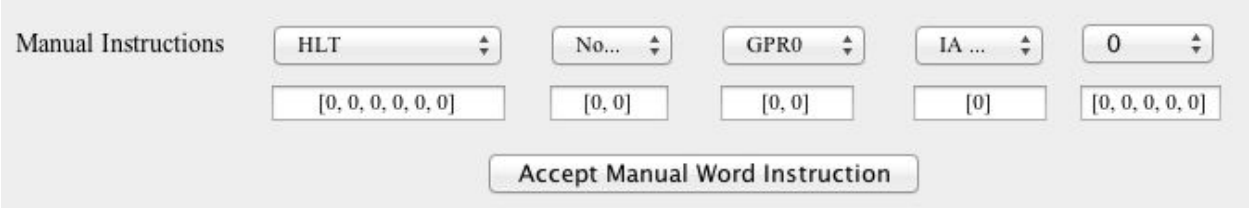
Part 3 - Demonstrate instructions through GUI(updated)

Deliverable description:

Load instructions from a file.

Explanation:

We are not intending to make duplicate load work, so keep having a list of opcode/assembly instructions (as phase 2) that the GUI can "load" from to replace load instructions from a file.

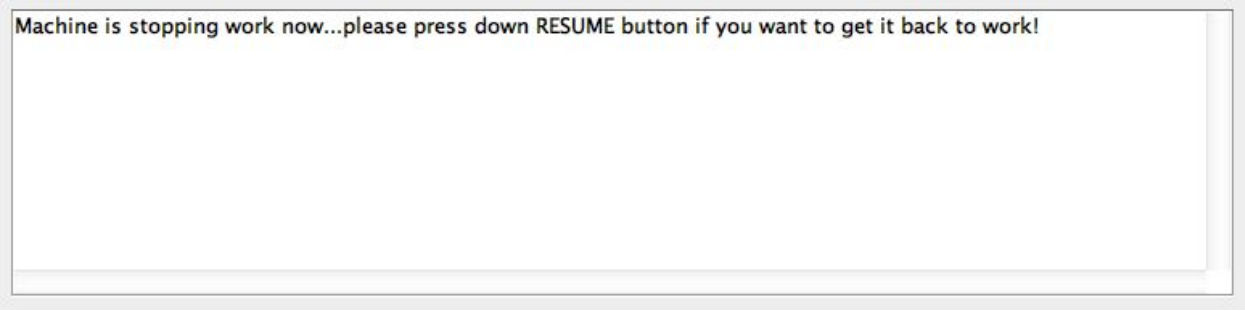


Demo Steps (Please reopen the GUI if the result is not expected):

1. HLT/RESUME

s1. Click on "HALT"

[Expected Result] -> ALU stops working, no more instructions will be executed.



s2. Click on "RESUME"

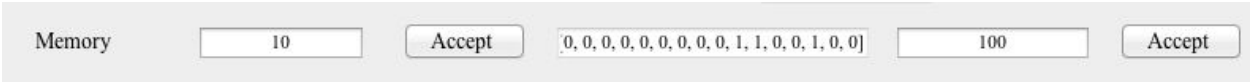
[Expected Result] -> ALU gets back to work.

2. LDR

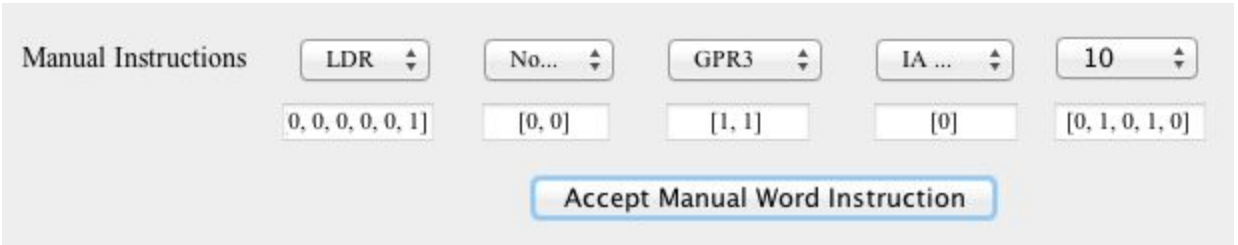
Case 1 - IX = 0, I = 0 -> EA = address

s1. input 10 to memory location -> accept

s2. input 100 to memory location's value -> accept



s3. select relevant values for LDR (IX = 0; GPR = 3; I = 0; Address = 10) in manual input part -> accept manual word instruction



[Expected Result] -> GPR3 is populated with 100

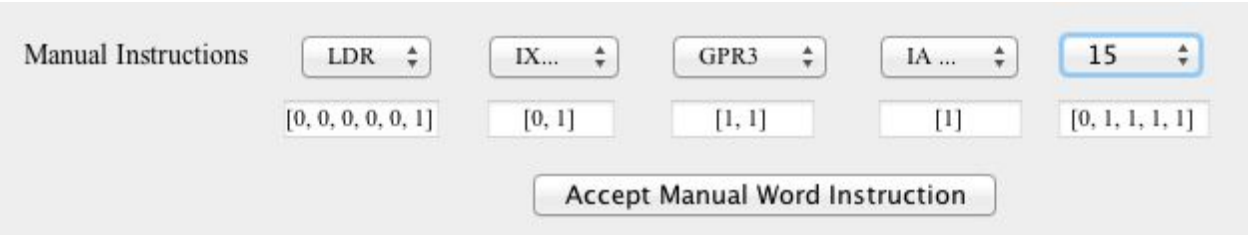
Case 2 - IX != 0, I = 1 -> EA = address + c(IR)

s1. input 5 to IX1

s2. input 20 to memory location -> accept

s3. input 100 to memory location's value -> accept

s4. select relevant values for LDR (IX = 1; GPR = 3; I = 1; Address = 15) in manual input part -> accept manual word instruction



[Expected Result] -> GPR3 is populated with 100

3. STR

Case 1 - IX = 0, I = 0 -> EA = address

s1. input 100 to GPR3 -> accept

s2. select relevant values for STR (IX = 0; GPR = 3; I = 0; Address = 20) in manual input part -> accept manual word instruction

[Expected Result] -> 100 is stored into memory address 20 (please click Accept close to Memory[20] to get content[100] visible)

Case 2 - IX != 0, I = 1 -> EA = address + c(IR)

s1. input 5 to IX1

s2. input 100 to GPR3 -> accept

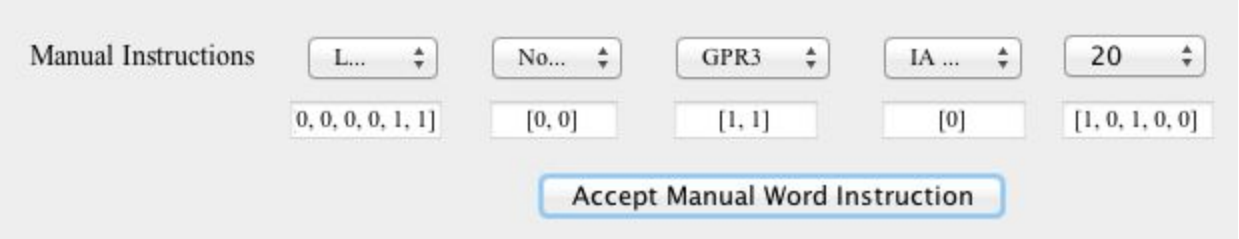
s3. select relevant values for STR (IX = 1; GPR = 3; I = 1; Address = 15) in manual input part -> accept manual word instruction

[Expected Result] -> 100 is stored into memory address 20 (please click Accept close to Memory[20] to get content[100] visible)

4. LDA

Case 1 - IX = 0, I = 0 -> EA = address

s1. select relevant values for LDA (IX = 0; GPR = 3; I = 0; Address = 20) in manual input part -> accept manual word instruction



[Expected Result] -> GPR3 is populated with 20

Case 2 - IX != 0, I = 1 -> EA = address + c(IR)

s1. input 5 to IX1

s2. select relevant values for LDA (IX = 1; GPR = 3; I = 1; Address = 20) in manual input part -> accept manual word instruction

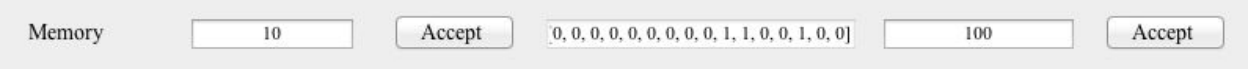
[Expected Result] -> GPR3 is populated with 25

5. LDX

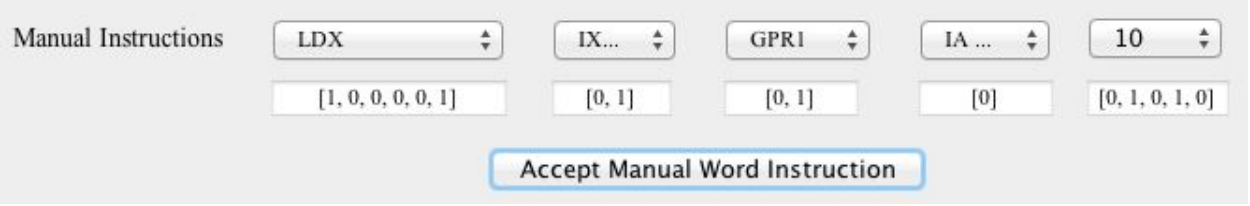
(Please note here the Index Register and GPR should be worked in pair, IX1 works with GPR1, IX2 works with GPR2 and IX3 works with GPR3)

s1. input 10 to memory location -> accept

s2. input 100 to memory location's value -> accept



s3. select relevant values for LDX (IX = 1; GPR = 1; I = 0; Address = 10) in manual input part -> accept manual word instruction



[Expected Result] -> IX1 is populated with 100

6. STX

s1. repeat 5. LDX

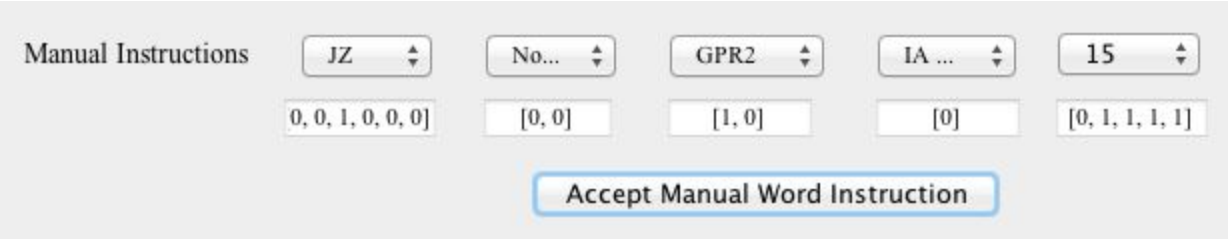
s2. select relevant values for STX (IX = 1; GPR = 3; I = 0; Address = 31) in manual input part -> accept manual word instruction

[Expected Result] -> 100 is stored into memory location 31 (please click Accept close to Memory[31] to get content[100] visible)

7. JZ

Case 1 - If c(r) = 0, then PC <− EA(address)

s1. select relevant values for JZ (IX = 0; GPR = 2; I = 0; Address = 15) in manual input part -> accept manual word instruction



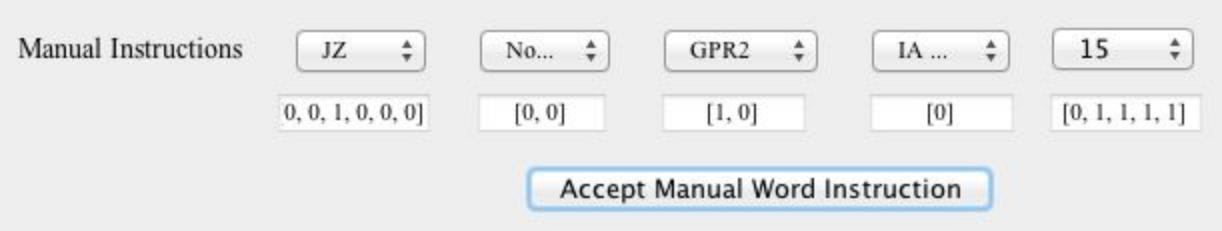
[Expected Result] -> PC is populated with address 15

Case 2 - If c(r) != 0, then PC <– PC + 1

s1. input 10 to GPR2

s2. PC is populated with 15

s3. select relevant values for JZ (IX = 0; GPR = 2; I = 0; Address = 15)
in manual input part -> accept manual word instruction



[Expected Result] -> PC is populated with address 16

8. JNE

Case 1 - If c(r) != 0, then PC <– EA(address)

s1. input 100 to GPR0

s2. select relevant values for JNE (IX = 0; GPR = 0; I = 0; Address = 15)
in manual input part -> accept manual word instruction

[Expected Result] -> PC is populated with address 15

Case 2 - If c(r) != 0, IX != 0 and I = 1 then PC <– address + c(IX)

s1. input 100 to GPR0

s2. input 5 to IX1

s3. select relevant values for JNE (IX = 1; GPR = 0; I = 1; Address = 15) in manual input part -> accept manual word instruction

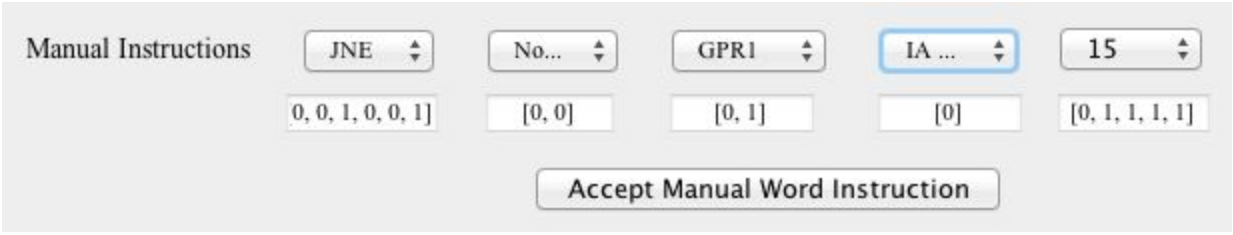
[Expected Result] -> PC is populated with address 20

Case 3 - If c(r) = 0, then PC <– PC + 1

s1. input 0 to GPR1

s2. PC is populated with 20

s3. select relevant values for JNE (IX = 0; GPR = 1; I = 0; Address = 15) in manual input part -> accept manual word instruction



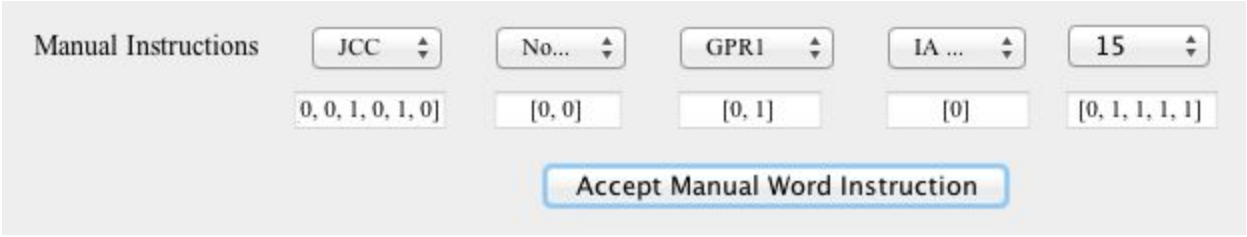
[Expected Result] -> PC is populated with 21

9. JCC

Case 1 - If cc bit = 1, PC <– EA(address)

s1. input 15 to Condition Code

s2. select relevant values for JCC (IX = 0; GPR = 1; I = 0; Address = 15) in manual input part -> accept manual word instruction



[Expected Result] -> PC is populated with 15

Case 2 - If cc bit = 0, PC <– PC + 1

s1. input 15 to PC

s2. input 0 to Condition Code

s3. select relevant values for JCC (IX = 0; GPR = 1; I = 0; Address = 15)
in manual input part -> accept manual word instruction

[Expected Result] -> PC is populated with 16

10. JMA

Case 1 - PC <- EA(address), if I bit not set

s1. select relevant values for JMA (IX = 0; GPR = 1; I = 0; Address = 15)
in manual input part -> accept manual word instruction

[Expected Result] -> PC is populated with 15

Case 2 - PC <- Address + c(IX), if I bit set

s1. input 5 to IX1

s2. select relevant values for JMA (IX = 1; GPR = 1; I = 1; Address = 15)

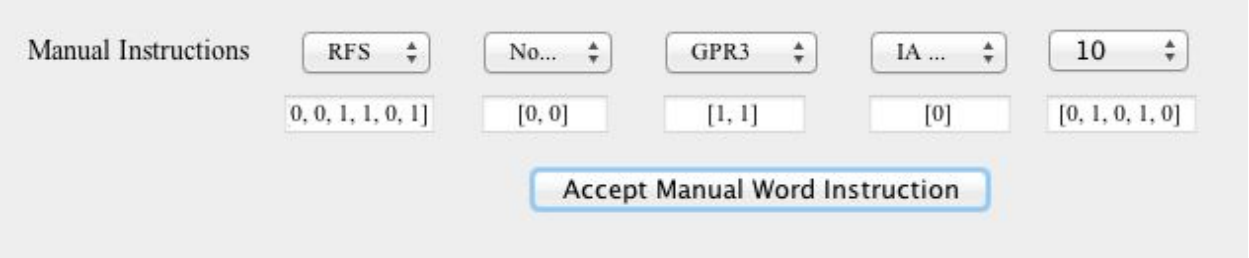
in manual input part -> accept manual word instruction

[Expected Result] -> PC is populated with 20

11. RFS - R0 <– Immed; PC <– c(R3)

s1. input 20 to GPR3

s2. select relevant values for RFS (IX = 0; GPR = 3; I = 0; Address = 10)
in manual input part -> accept manual word instruction



[Expected Result] -> GPR0 is populated with 10, PC is populated with 20

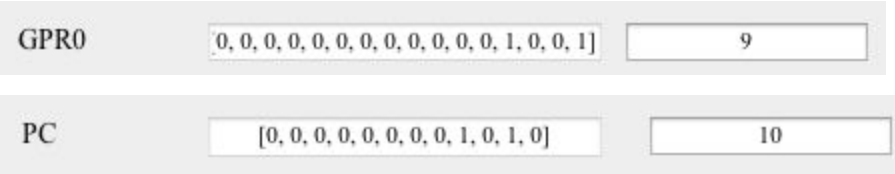
12. SOB

Case 1 - r <– c(r) – 1 If c(r) > 0, PC <- EA(address)

s1. input 10 to GPR0

s2. select relevant values for SOB (IX = 0; GPR = 0; I = 0; Address = 10) in manual input part -> accept manual word instruction

[Expected Result] -> GPR0 = 9; PC = 10

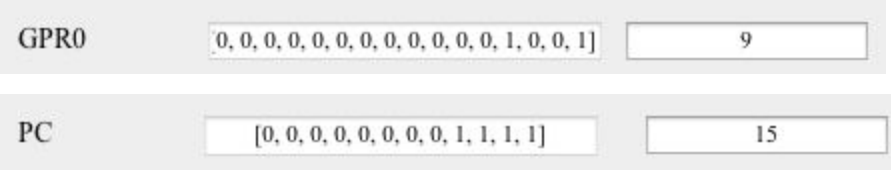


Case 2 - r <– c(r) – 1 If c(r) > 0, PC <- Address + c(IX) if IX != 0 and I = 1
s1. input 10 to GPR0

s2. input 5 to IX1

s3. select relevant values for SOB (IX = 1; GPR = 0; I = 1; Address = 10) in manual input part -> accept manual word instruction

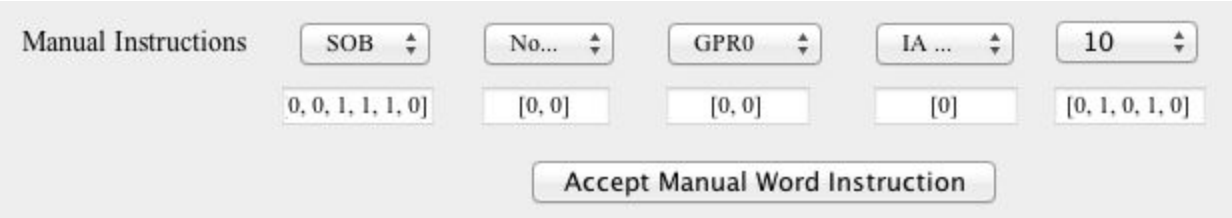
[Expected Result] -> GPR0 = 9; PC = 15



Case 3 - If c(r) = 0, PC <- PC + 1
s1. input 0 to GPR0

s2. input 20 to PC

s3. select relevant values for SOB (IX = 0; GPR = 0; I = 0; Address = 10)
in manual input part -> accept manual word instruction

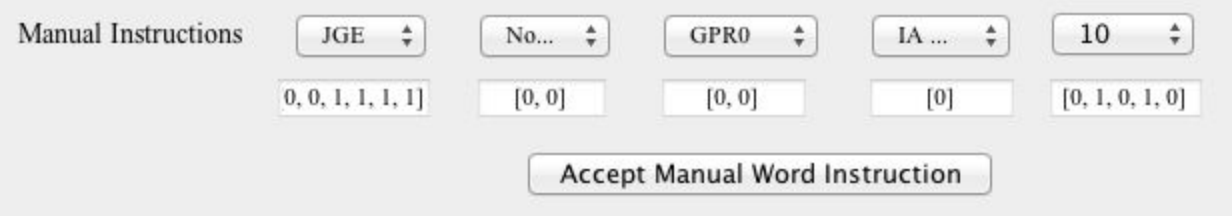


[Expected Result] -> PC = 21

13. JGE

Case 1 - If c(r) > = 0, then PC <- EA(address) if IX = 0 and I = 0
s1. input 10 to GPR0

s2. select relevant values for JGE (IX = 0; GPR = 0; I = 0; Address = 10)
in manual input part -> accept manual word instruction



[Expected Result] -> PC = 10

Case 2 - If c(r) > = 0, IX != 0 and I = 1, then PC <- Address + c(IX)
s1. input 5 to IX1

s2. input 10 to GPR0

s3. select relevant values for JGE (IX = 1; GPR = 0; I = 1; Address = 10)
in manual input part -> accept manual word instruction

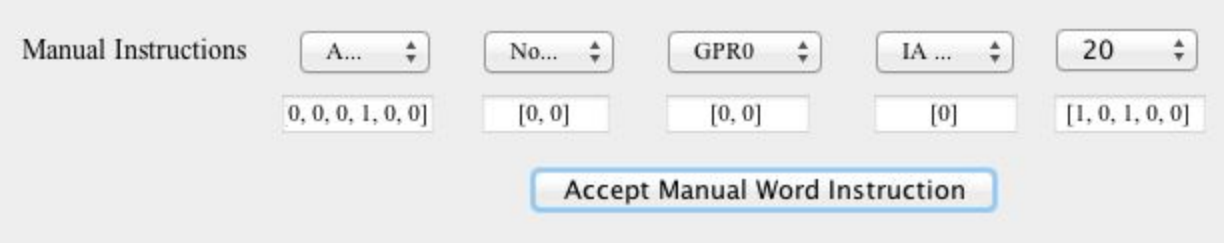
[Expected Result] -> PC = 15

14. AMR

s1. input 20 to memory location -> accept
s2. input 100 to memory location's value -> accept

s3. input 100 to GPR0

s4. select relevant values for AMR (IX = 0; GPR = 0; I = 0; Address = 20)
in manual input part -> accept manual word instruction



[Expected Result] -> GPR0 = 200

15. SMR

s1. input 200 to GPR0

s2. input 20 to memory location -> accept

s3. input 100 to memory location's value -> accept

s4. select relevant values for SMR (IX = 0; GPR = 0; I = 0; Address = 20)

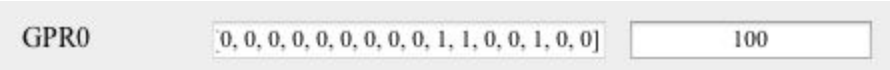
in manual input part -> accept manual word instruction

[Expected Result] -> GPR0 = 100

16. AIR

Case 1 - r <– c(r) + Immed, do nothing if Immed = 0

s1. input 100 to GPR0



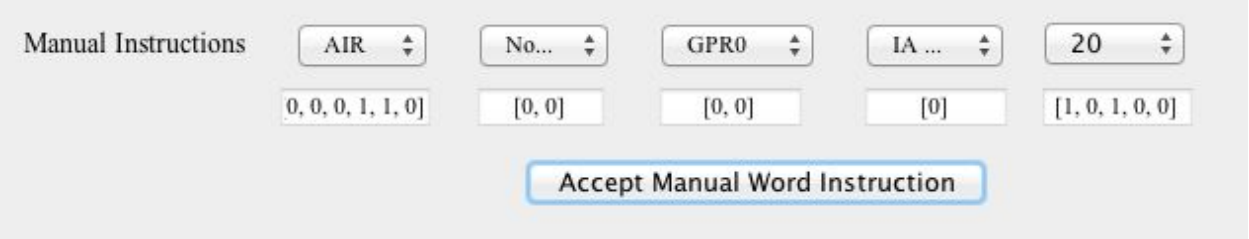
s2. select relevant values for AIR (IX = 0; GPR = 0; I = 0; Address = 0)

in manual input part -> accept manual word instruction

[Expected Result] -> GPR0 = 100

Case 2 - r <– c(r) + Immed, loads r with Immed, if c(r) = 0
s1. input 0 to GPR0

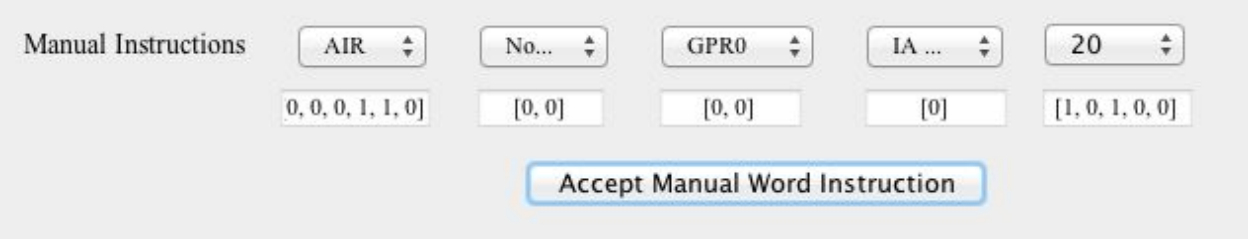
s2. select relevant values for AIR (IX = 0; GPR = 0; I = 0; Address = 20)
in manual input part -> accept manual word instruction



[Expected Result] -> GPR0 = 20

Case 3 - r <– c(r) + Immed
s1. input 100 to GPR0

s2. select relevant values for AIR (IX = 0; GPR = 0; I = 0; Address = 20)
in manual input part -> accept manual word instruction



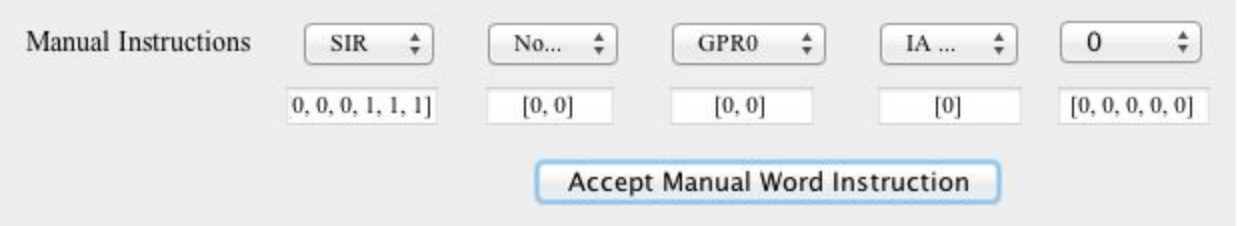
[Expected Result] -> GPR0 = 120

17. SIR

Case 1 - r <– c(r) - Immed, do nothing if Immed = 0

s1. input 100 to GPR0

s2. select relevant values for SIR (IX = 0; GPR = 0; I = 0; Address = 0)
in manual input part -> accept manual word instruction

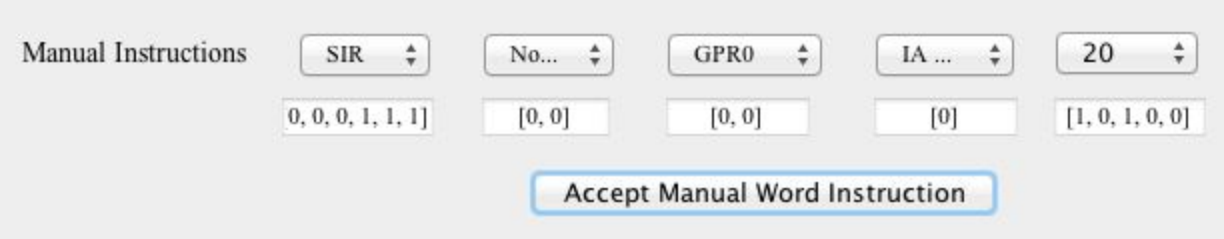


[Expected Result] -> GPR0 = 100

Case 2 - r <– c(r) - Immed

s1. input 100 to GPR0

s2. select relevant values for SIR (IX = 0; GPR = 0; I = 0; Address = 20)
in manual input part -> accept manual word instruction



[Expected Result] -> GPR0 = 80

18. NOT

s1. input 8 to GPR0

s2. select relevant values for NOT (IX = 0; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction

[Expected Result] -> GPR0 = ~8

19. SRC

(Index Register is used to decide left/right shift, address value is used to decide the bits to shift)

Case 1 - left shift

s1. input 10 to GPR0

s2. input 1 to IX1

s3. select relevant values for SRC (IX = 1; GPR = 0; I = 0; Address = 1) in manual input part -> accept manual word instruction

[Expected Result] -> GPR0 = 20

Case 2 - right shift

s1. input 20 to GPR0

s2. select relevant values for SRC (IX = 0; GPR = 0; I = 0; Address = 1) in manual input part -> accept manual word instruction

[Expected Result] -> GPR0 = 10

20. RRC

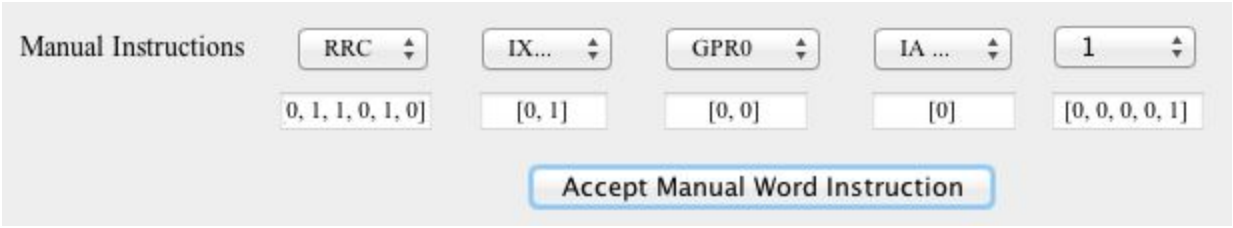
(Index Register is used to decide left/right shift, address value is used to decide the bits to shift)

Case 1 - left shift

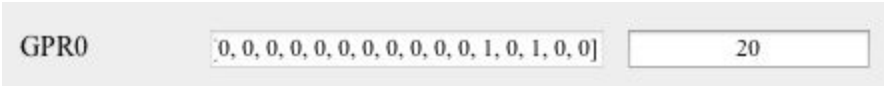
s1. input 10 to GPR0

s2. input 1 to IX1

s3. select relevant values for RRC (IX = 1; GPR = 0; I = 0; Address = 1) in manual input part -> accept manual word instruction



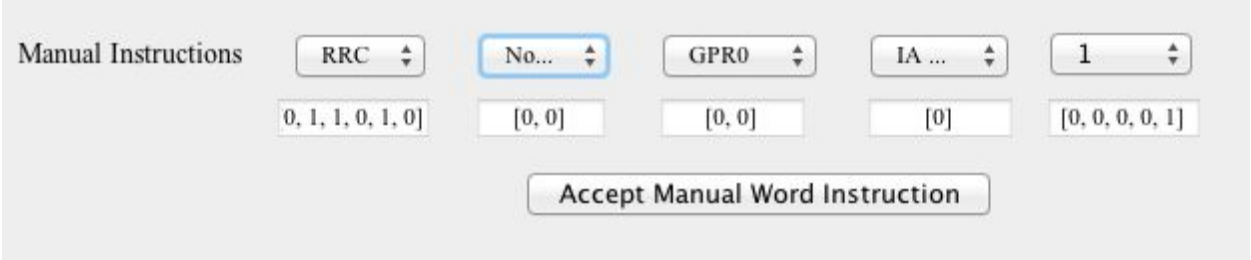
[Expected Result] -> GPR0 = 20



Case 2 - right shift

s1. input 10 to GPR0

s2. select relevant values for RRC (IX = 0; GPR = 0; I = 0; Address = 1) in manual input part -> accept manual word instruction



[Expected Result] -> GPR0 = 5

21. MLT

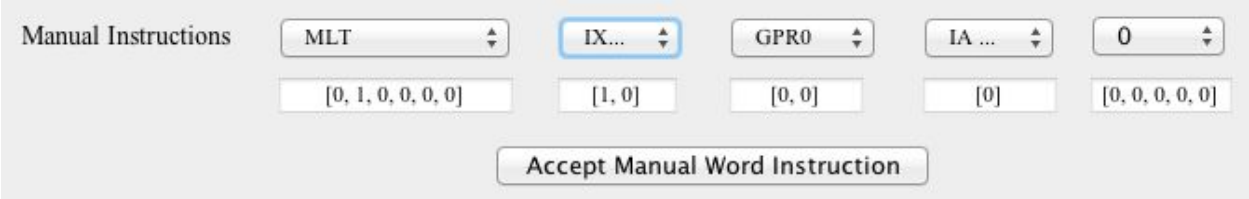
Case 1 - positive * positive

s1. input 10 to GPR0

s2. input 10 to GPR2

s3. input 0 to IX2

s4. select relevant values for MLT (**IX = 2[have to use this]**; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction



[Expected Result] -> GPR1 is populated with 100 (10*10)

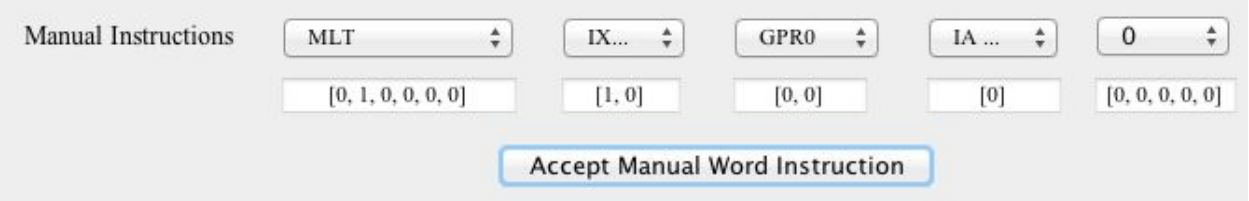
Case 2 - positive * negative

s1. input 10 to GPR0

s2. input -10 to GPR2

s3. input 0 to IX2

s4. select relevant values for MLT (**IX = 2[have to use this]**; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction



[Expected Result] -> GPR1 is populated with -100 (10*-10)

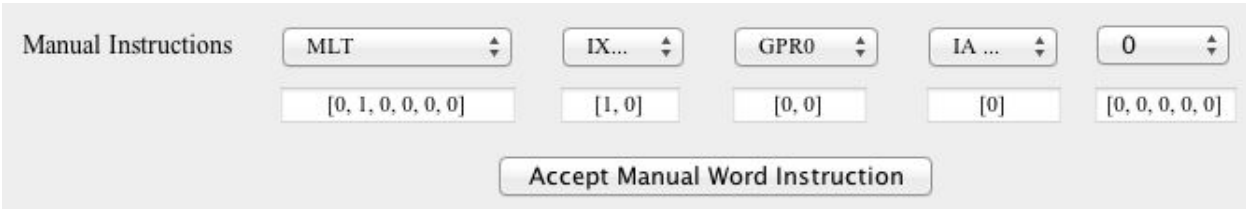
Case 3 - overflow (the result should be less than 65536)

s1. input 32700 to GPR0

s2. input 2 to GPR2

s3. input 0 to IX2

s4. select relevant values for MLT (**IX = 2[have to use this]**; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction



[Expected Result] -> CC is populated with 1 (0001)

22. DVD

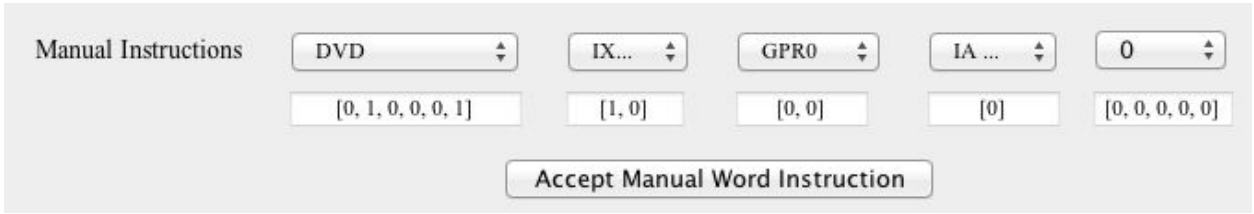
Case 1 - positive / positive

s1. input 100 to GPR0

s2. input 10 to GPR2

s3. input 0 to IX2

s4. select relevant values for DVD (**IX = 2[have to use this]**; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction



[Expected Result] -> GPR0 is populated with 10 (100/10)

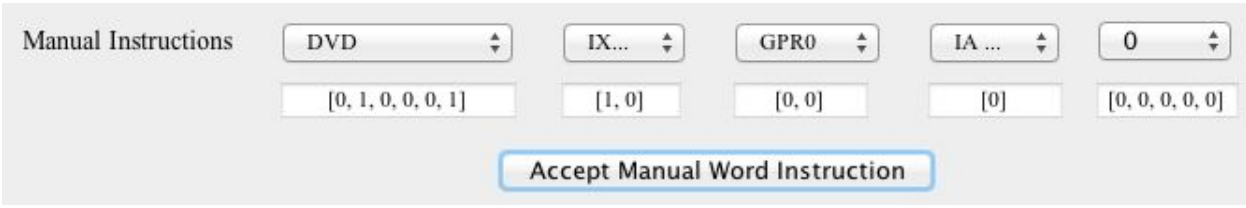
Case 2 - positive / negative

s1. input 100 to GPR0

s2. input -2 to GPR2

s3. input 0 to IX2

s4. select relevant values for DVD (**IX = 2[have to use this]**; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction



[Expected Result] -> GPR0 is populated with -50 (100/-2)

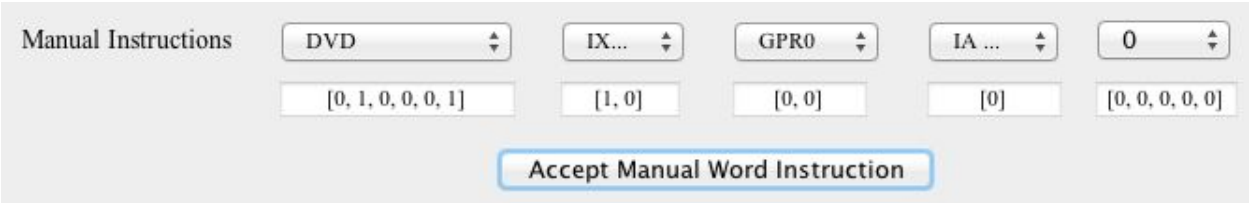
Case 3 - division by zero

s1. input 100 to GPR0

s2. input 0 to GPR2

s3. input 0 to IX2

s4. select relevant values for DVD (**IX = 2[have to use this]**; GPR = 0; I = 0; Address = 0) in manual input part -> accept manual word instruction



[Expected Result] -> CC is populated with 4 (0100)

23. JSR/TRR/AND/ORR/IN/OUT

Our test program 1 and 2 have covered these SIX instructions, so not intended to demonstrate them again through this step.

End of Document