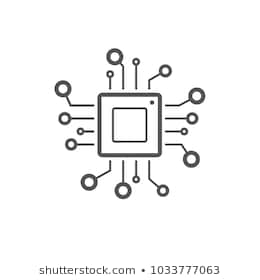
­­­­

Atma Ram Sanatan Dharma College  
University of Delhi



Computer System Architecture

Practical File for Paper Code 32341102

Submitted By  
Sudipto Ghosh  
College Roll No. 19/78003  
BSc (Hons) Computer Science

Submitted To  
Ms Uma Ojha   
Department of Computer Science

# INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| S No. | Objective | Date | Sign |
| 1 | Create a machine based on the given architecture and design the register set, memory and instruction formats for the machine | 03/09/19 |  |
| 2 | Create the micro-operations and associate with instructions as given in the list of basic computer instructions. Design the instruction set for the machine | 03/09/19 |  |
| 3 | Create the fetch routine for the basic computer as designed in the previous practical | 10/09/19 |  |
| 4 | Simulate the machine to determine the contents of AC, E, PC, AR, and IR registers in hexadecimal after the execution of given register reference instructions | 24/09/19 |  |
| 5 | Simulate the machine for the given memory-reference instructions with I = 0 and determine the contents of AC, E, PC, AR, and IR registers in hexadecimal after the execution | 24/09/19 |  |
| 6 | Simulate the machine for the given memory-reference instructions with I = 1 and determine the contents of AC, E, PC, AR, and IR registers in hexadecimal after the execution | 24/09/19 |  |
| 7 | Modify the machine created in Practical 1 according to the given instruction format | 01/10/19 |  |

# PRACTICAL 1

Dated 03rd September 2019

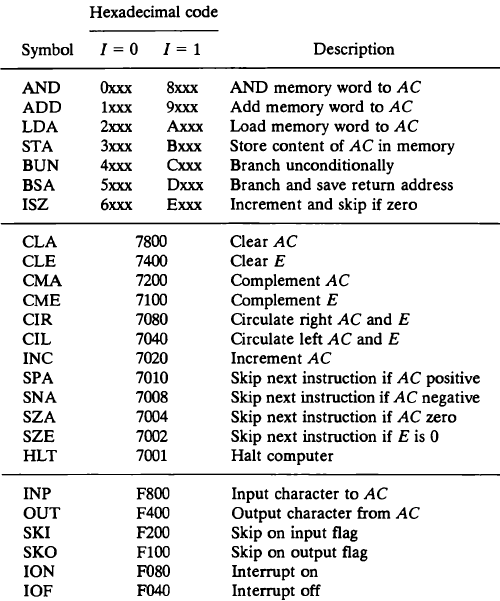
## Objective

Create a machine based on the given architecture and design the register set, memory and instruction formats for the machine.

### Register Set

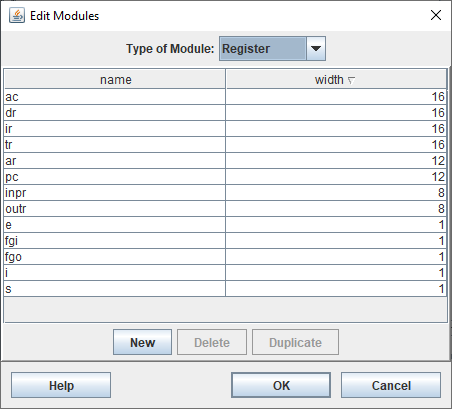
### Memory and Instruction Format

### Basic Computer Instructions

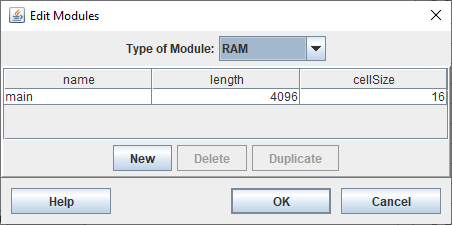


## Observation

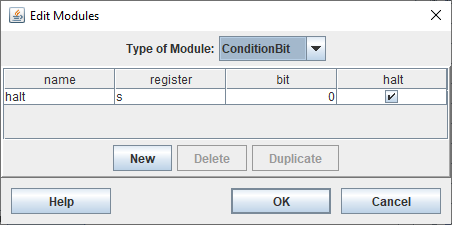
### Registers



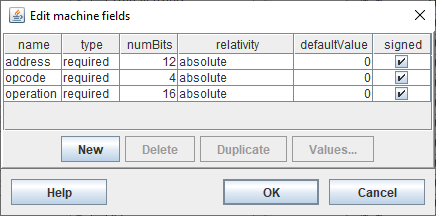
### Memory



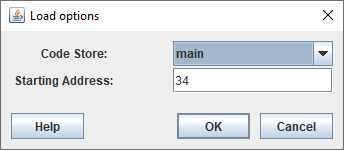
### Condition Bits



### Instruction Formats



### Loading Point



# PRACTICAL 2

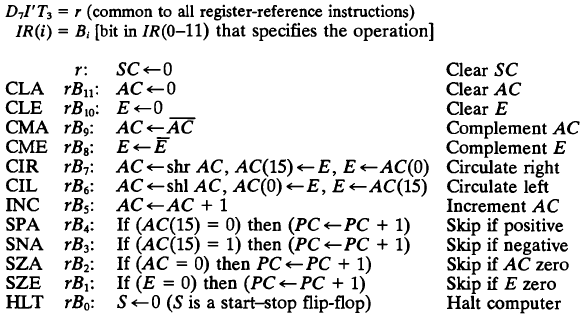
Dated 03rd September 2019

## Objective

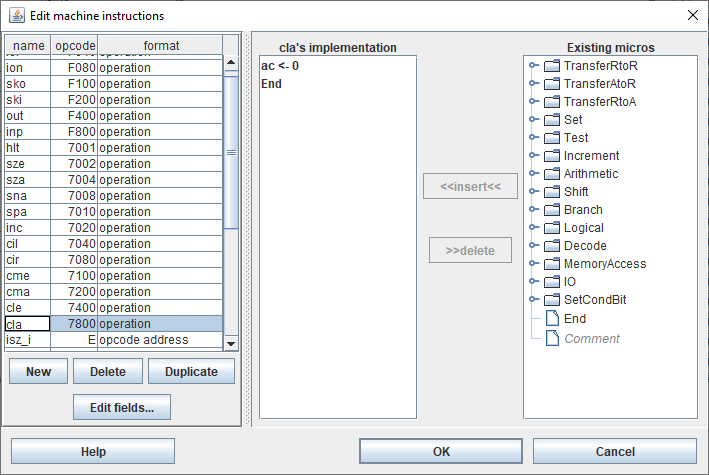
Create the micro-operations and associate with instructions as given in the list of basic computer instructions. Design the instruction set for the machine keeping in mind that bits are left indexed in CPUSim.

## Observation

### Register Reference Instructions

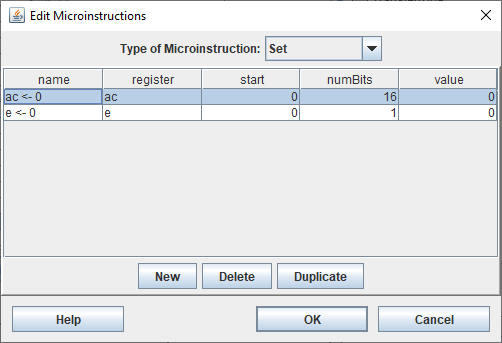


#### CLA – Clear AC



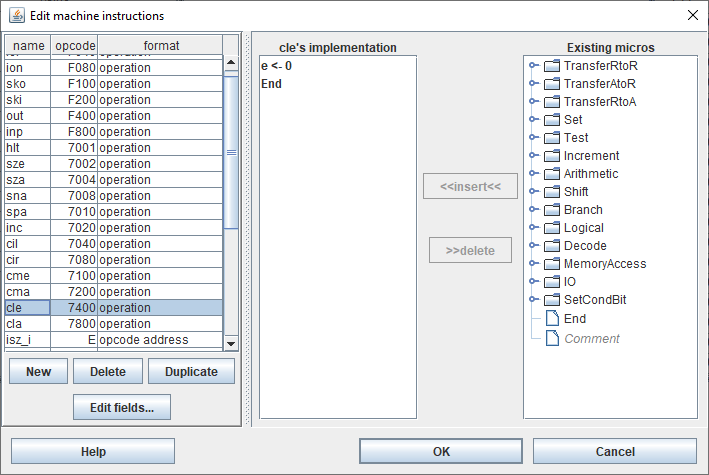
##### Implementation of CLA

* **AC <- 0**

****

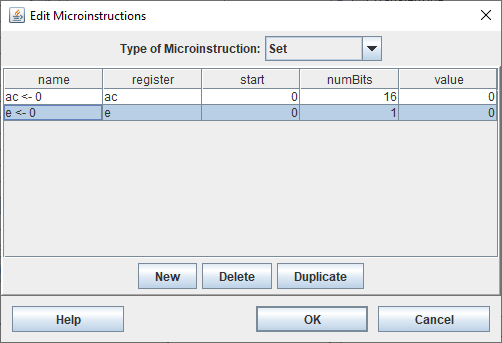
* **SC <- 0**

#### **CLE – Clear E**



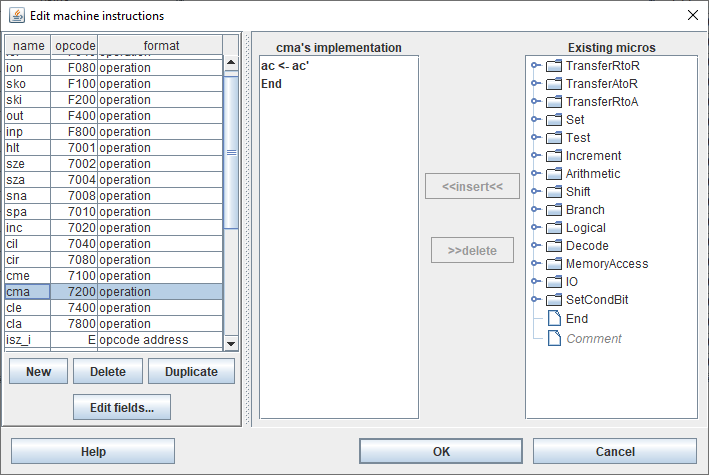
##### Implementation of CLE

* **E <- 0**



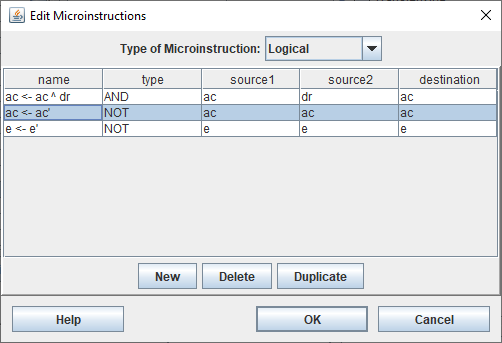
* **SC <- 0**

#### CMA – Complement AC



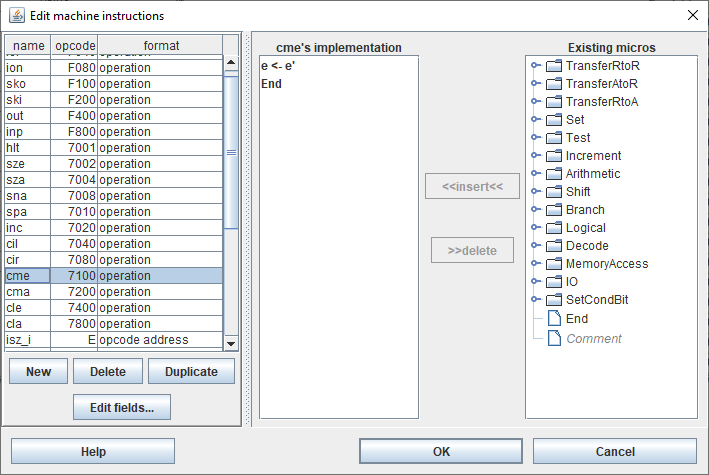
##### Implementation of CMA

* **AC <- AC ’**



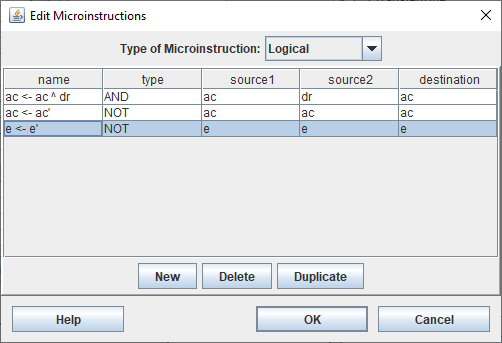
* **SC <- 0**

#### CME – Complement E



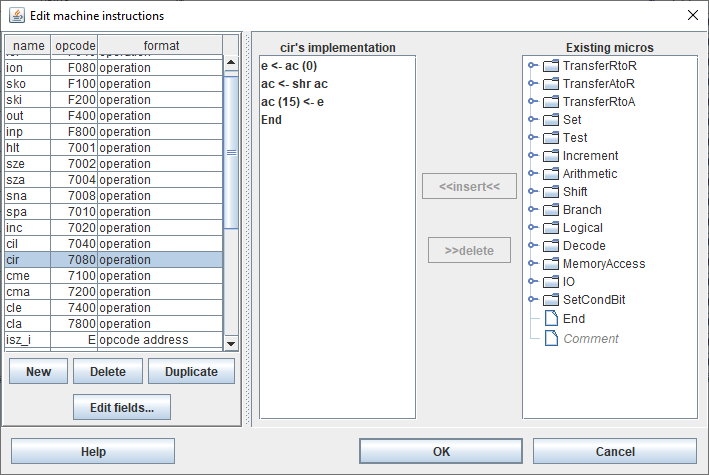
##### Implementation of CME

* **E <- E ‘**



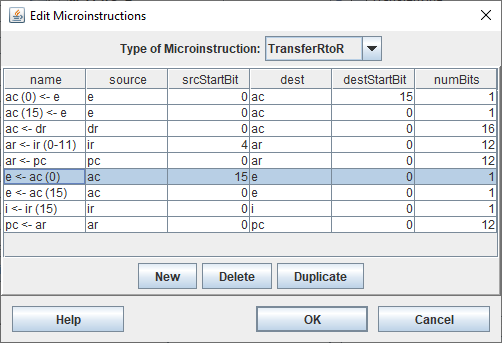
* **SC <- 0**

#### CIR – Circulate Right

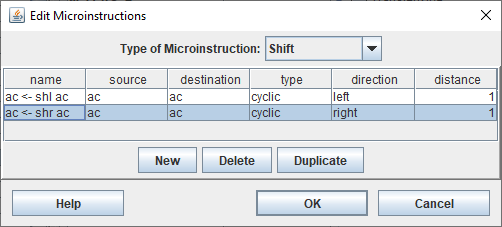


##### Implementation of CIR

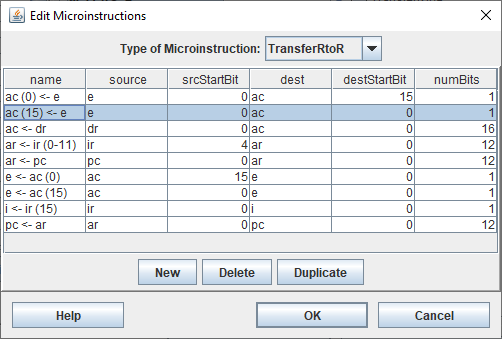
* **E <- AC (0)**



* **AC <- shr AC**

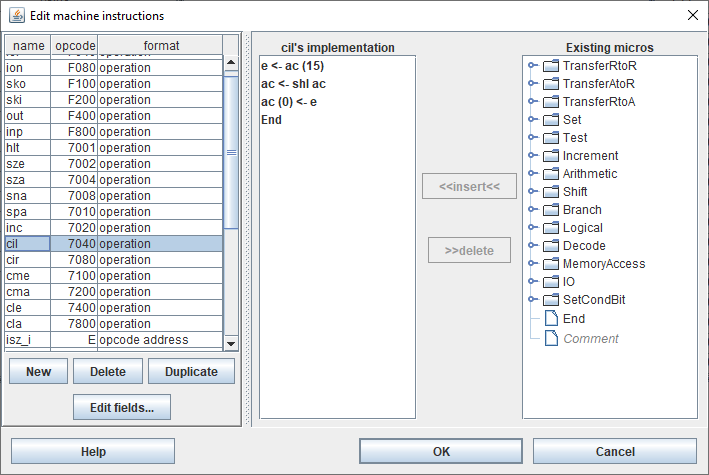


* **AC (15) <- 0**



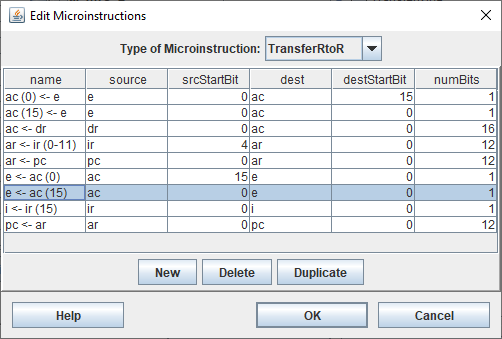
* **SC <- 0**

#### CIL – Circulate Left

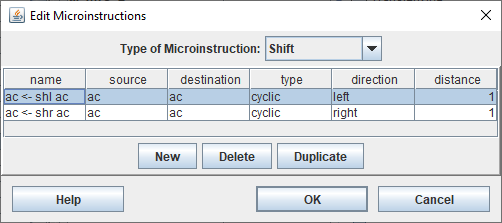


##### Implementation of CIL

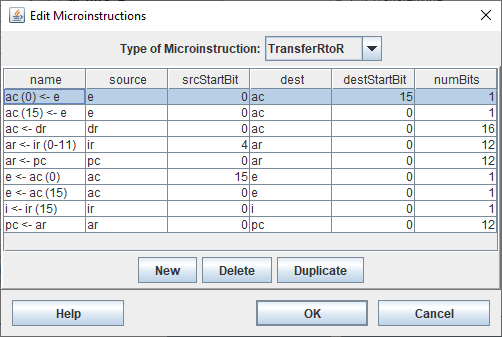
* **E <- AC (15)**



* **AC <- shl AC**

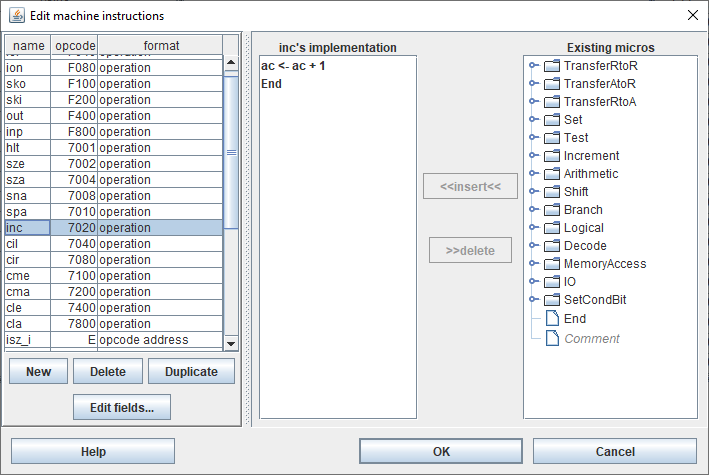


* **AC (0) <- E**



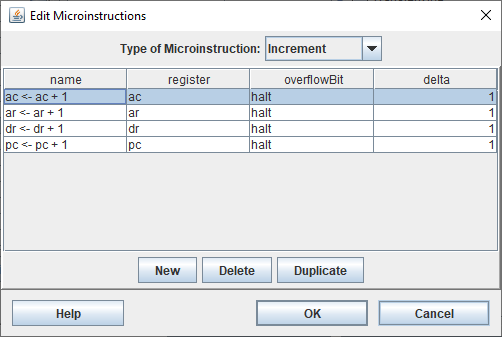
* **SC <- 0**

#### INC – Increment AC



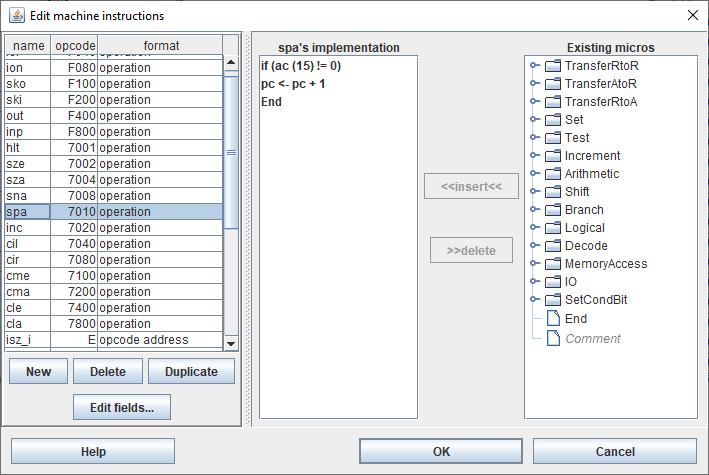
##### Implementation of INC

* **AC <- AC + 1**



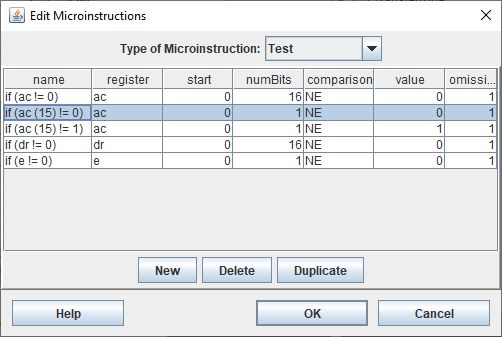
* **SC <- 0**

#### SPA – Skip If Positive

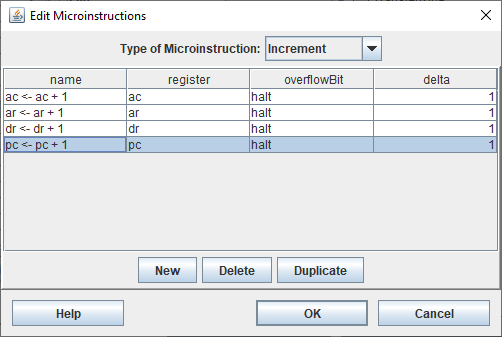


##### Implementation of SPA

* **If AC (15) != 0** (CPUSim skips a micro-operation when this is true)

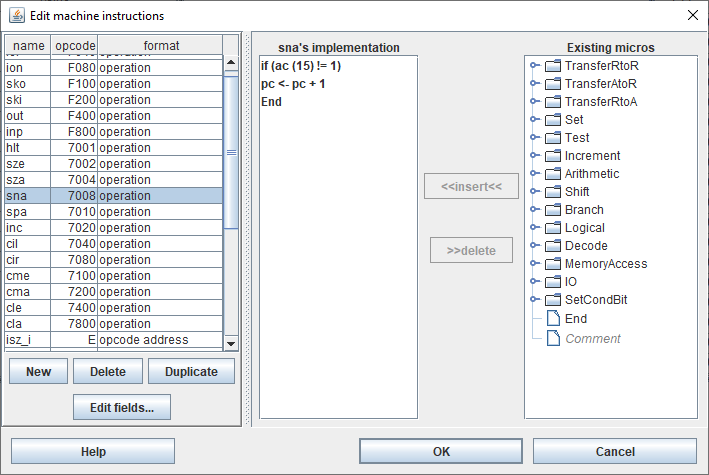


* **PC <- PC + 1**



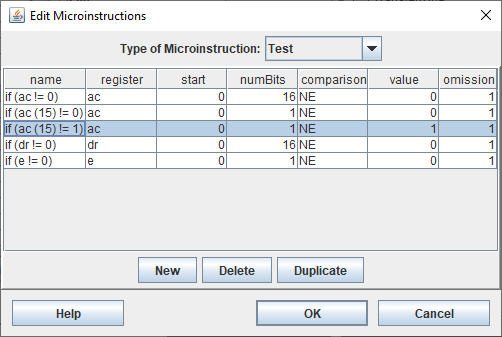
* **SC <- 0**

#### SNA – Skip If Negative

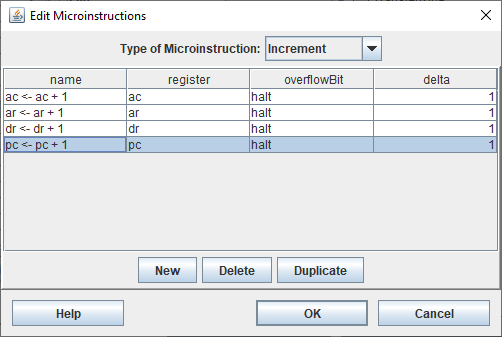


##### Implementation of SNA

* **If AC (15) != 1** (CPUSim skips a micro-operation when this is true)

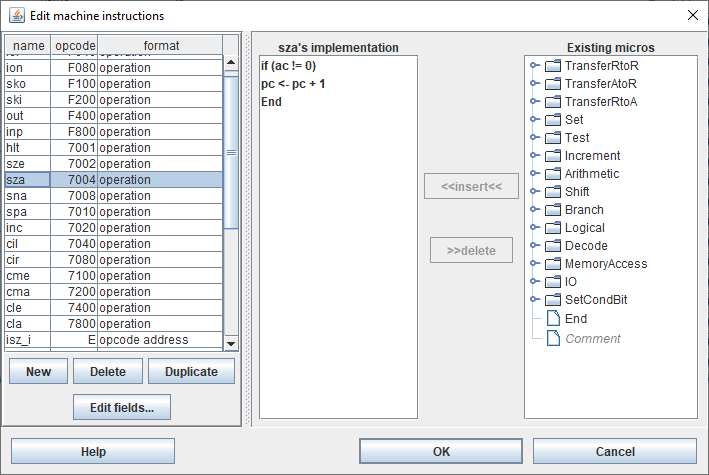


* **PC <- PC + 1**



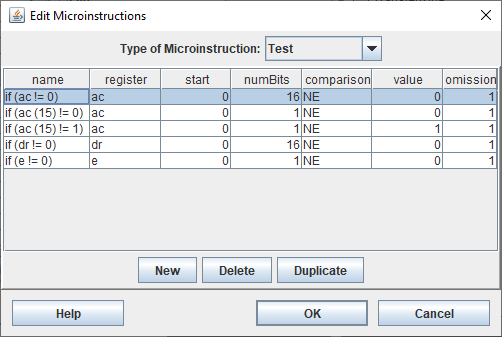
* **SC <- 0**

#### SZA – Skip If AC Zero

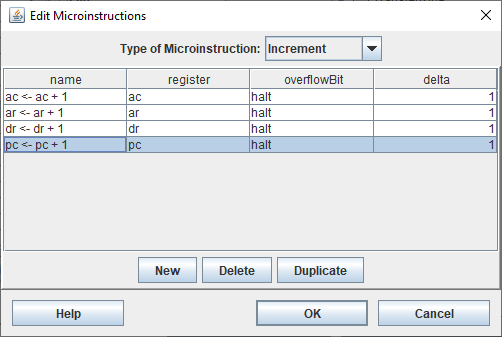


##### Implementation of SZA

* **If AC != 0** (CPUSim skips a micro-operation when this is true)

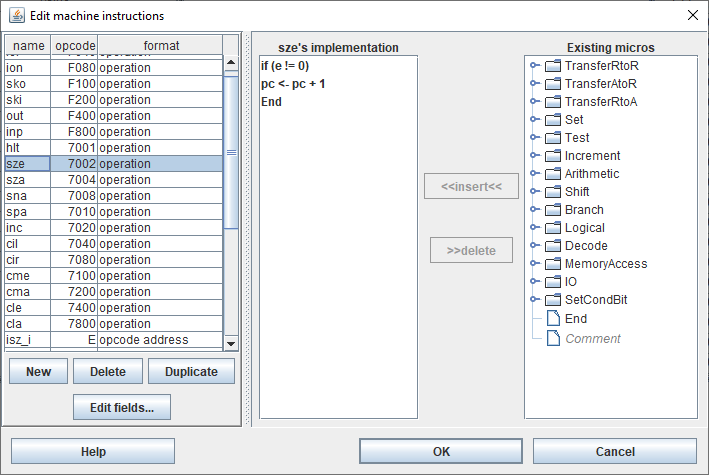


* **PC <- PC + 1**



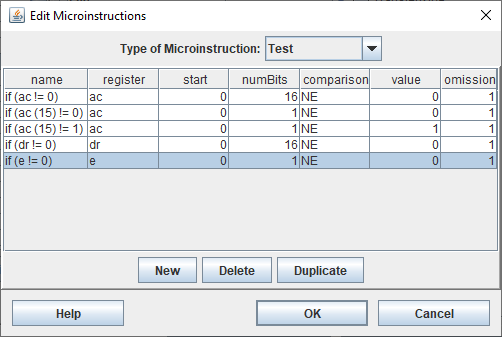
* **SC <- 0**

#### SZE – Skip If E Zero

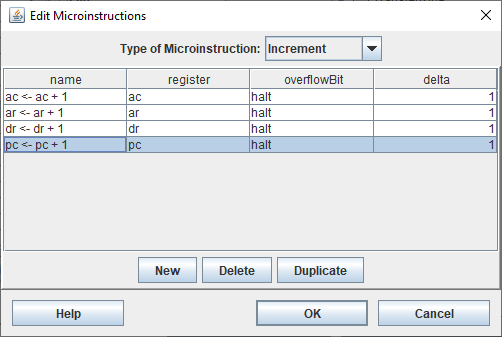


##### Implementation of SZE

* **If E != 0** (CPUSim skips a micro-operation when this is true)

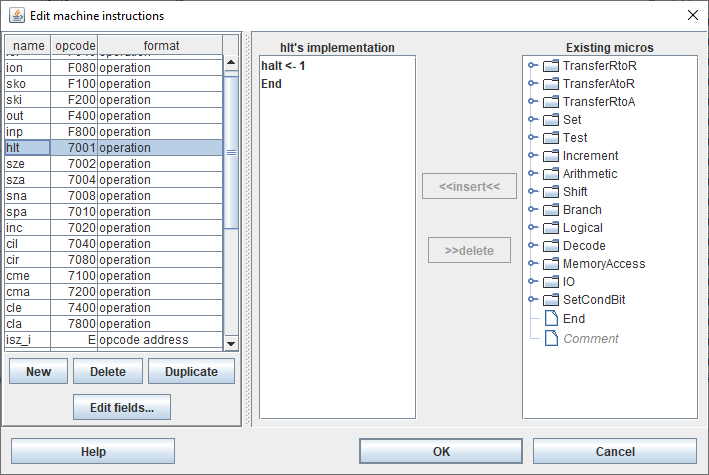


* **PC <- PC + 1**



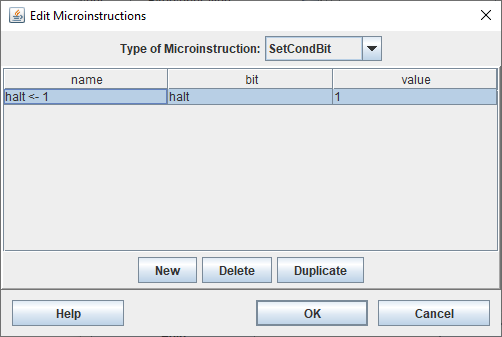
* **SC <- 0**

#### HLT – Halt Computer



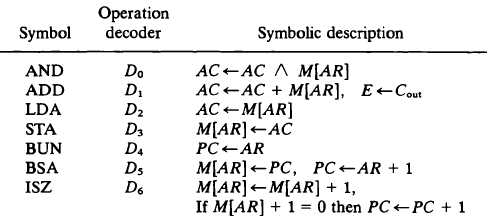
##### Implementation of HLT

* **HALT <-1 (S <- 1)**



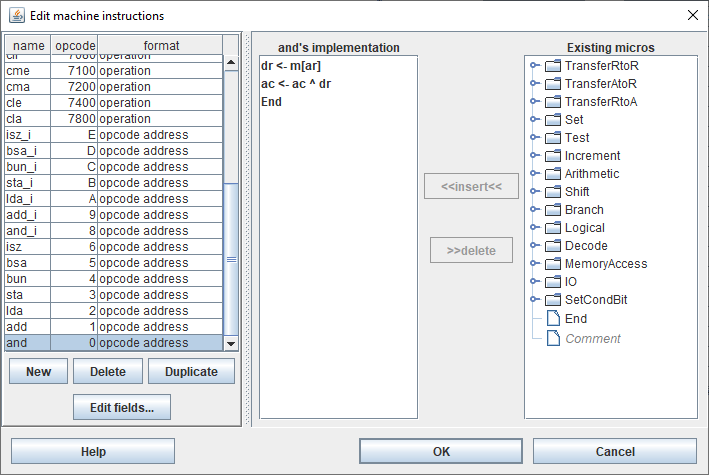
* **SC <- 0**

### Memory Reference Instructions



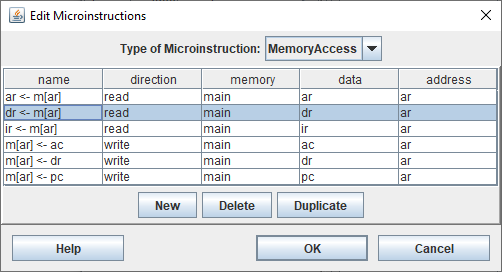
#### AND to AC

##### AND – Direct Addressing Mode

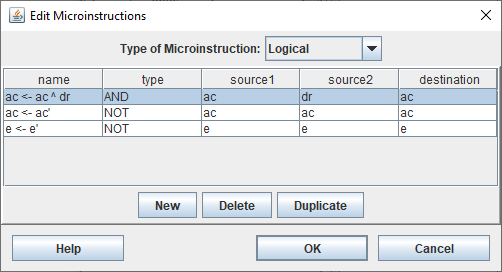


###### Implementation of AND

* **DR <- M[AR]**

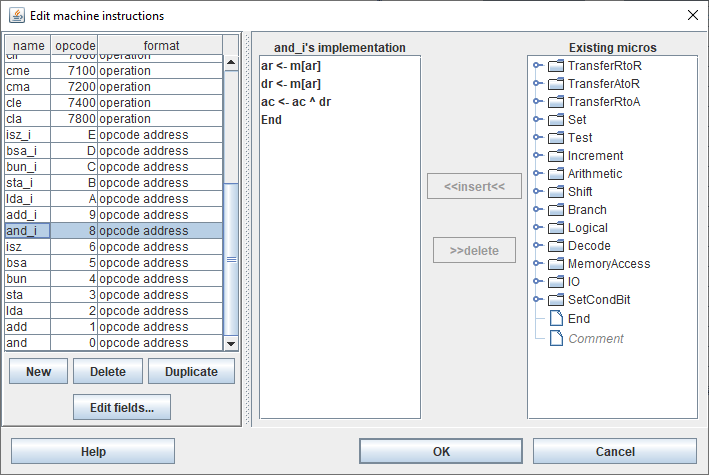


* **AC <- AC ^ DR**



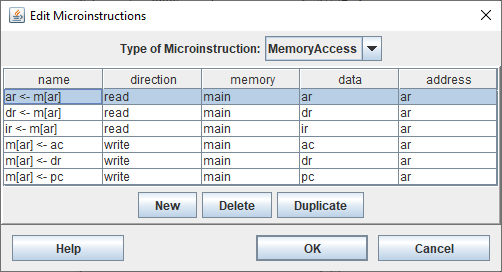
* **SC <- 0**

##### AND\_I – Indirect Addressing Mode

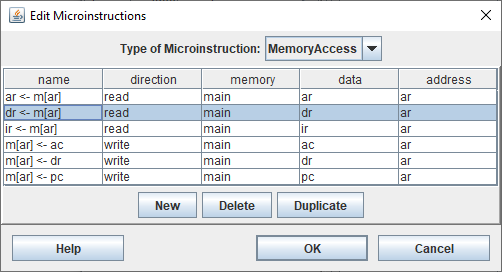


###### Implementation of AND\_I

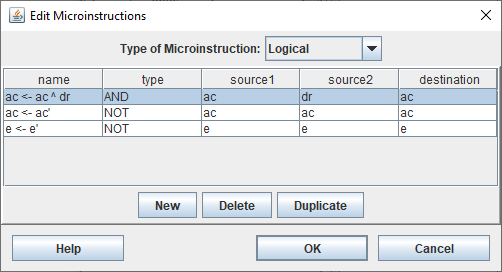
* **AR <- M[AR]**



* **DR <- M[AR]**



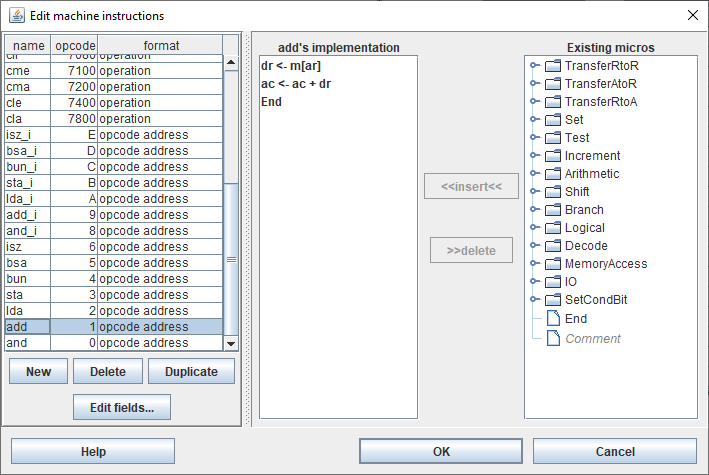
* **AC <- AC ^ DR**



* **SC <- 0**

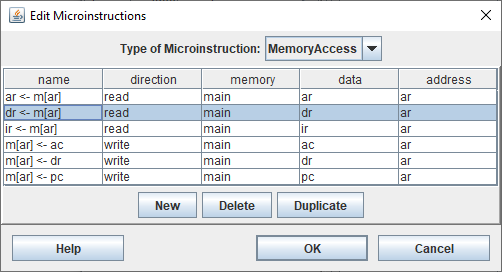
#### ADD to AC

##### ADD – Direct Addressing Mode

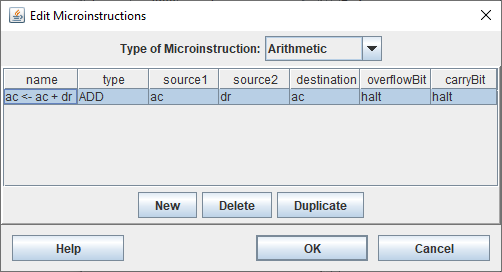


###### Implementation of ADD

* **DR <- M[AR]**

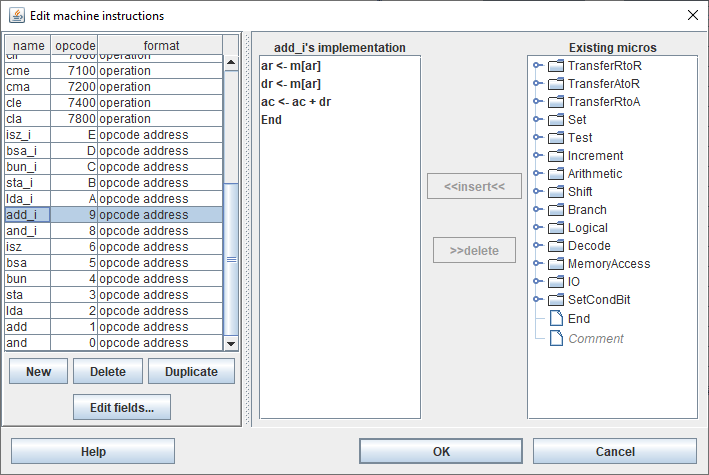


* **AC <- AC + DR**



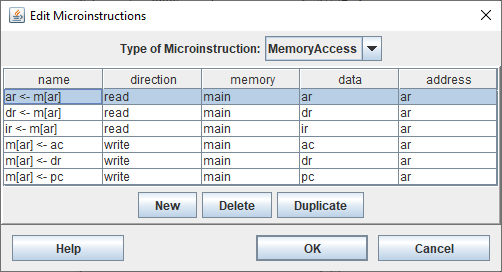
* **SC <- 0**

##### ADD\_I – Indirect Addressing Mode

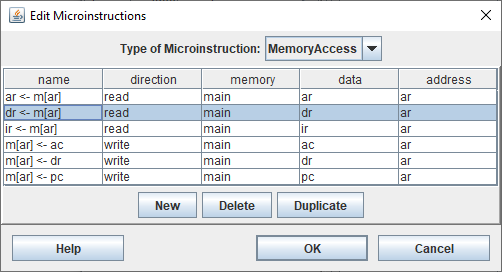


###### Implementation of ADD\_I

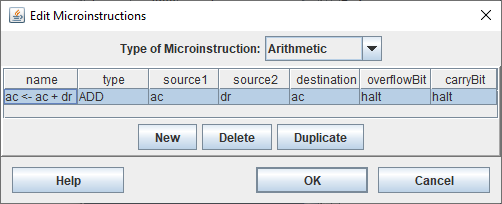
* **AR <- M[AR]**



* **DR <- M[AR]**



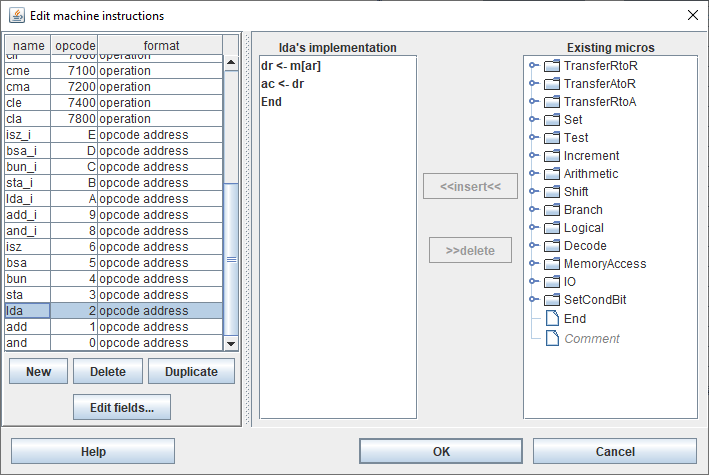
* **AC <- AC + DR**



* **SC <- 0**

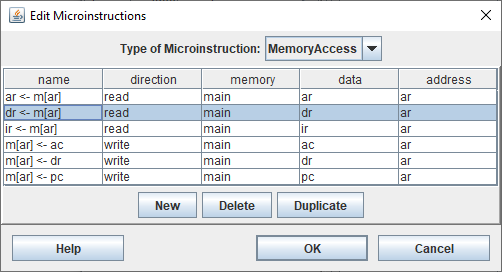
#### LOAD to AC

##### LDA – Direct Addressing Mode

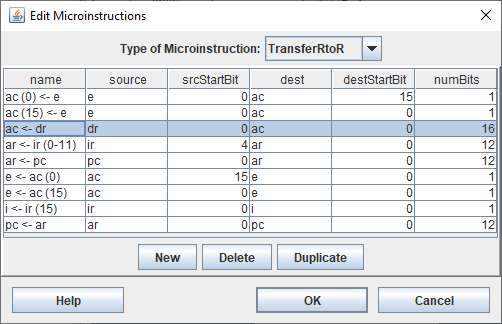


###### Implementation of LDA

* **DR <- M[AR]**

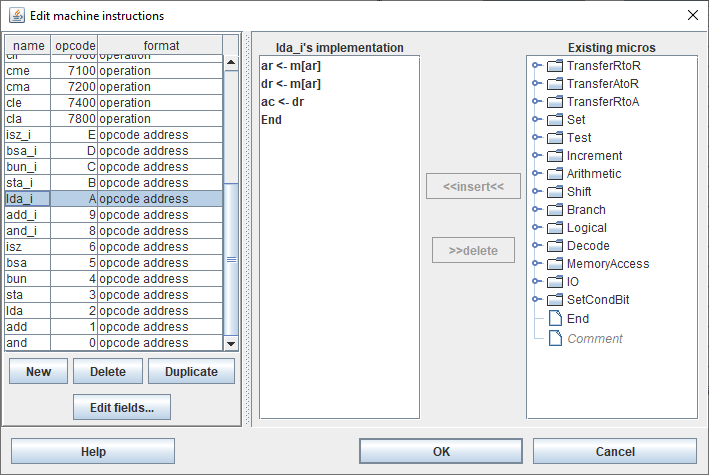


* **AC <- DR**



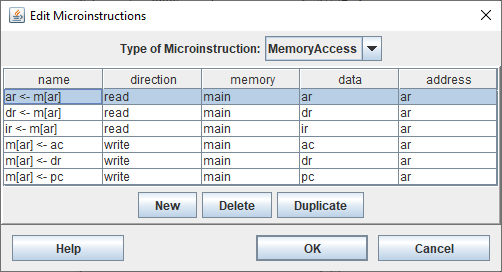
* **SC <- 0**

##### LDA\_I – Indirect Addressing Mode

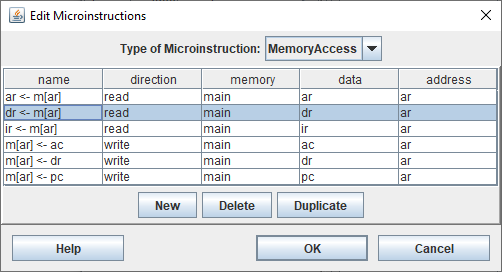


###### Implementation of LDA\_I

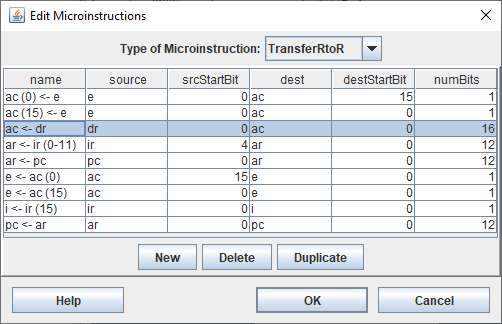
* **AR <- M[AR]**



* **DR <- M[AR]**



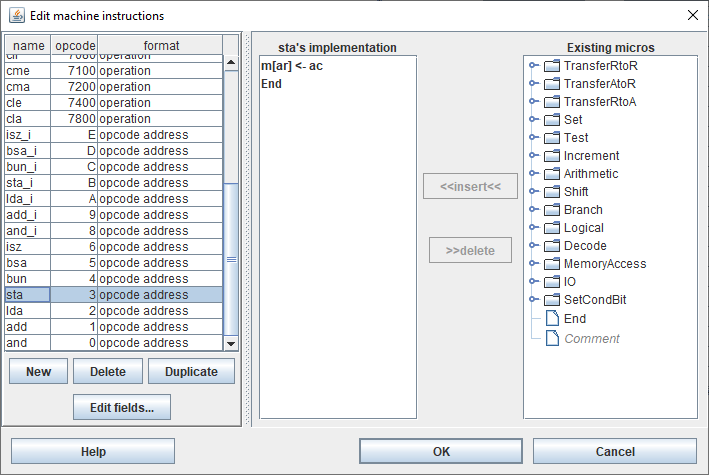
* **AC <- DR**



* **SC <- 0**

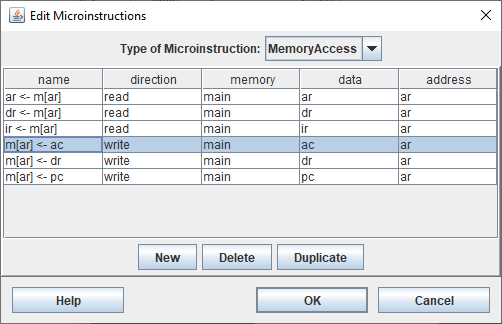
#### STORE AC

##### STA – Direct Addressing Mode



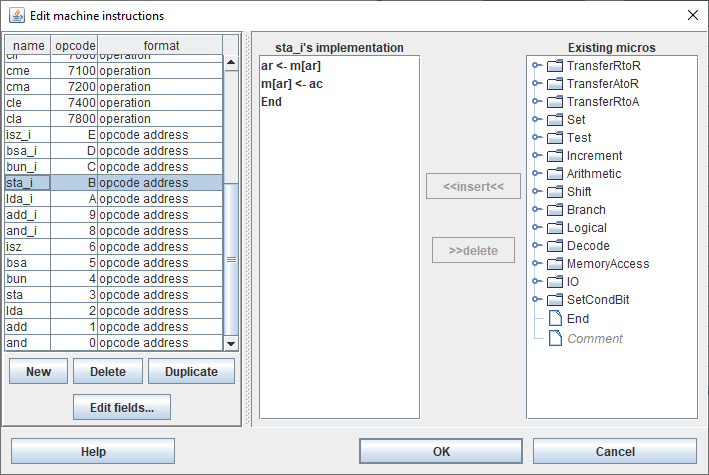
###### Implementation of STA

* **M[AR] <- AC**



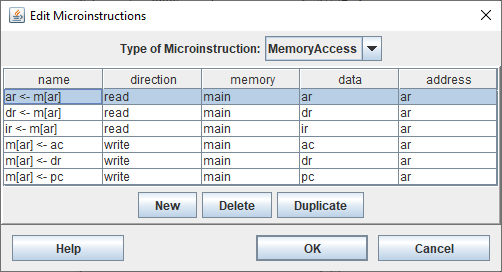
* **SC <- 0**

##### STA\_I – Indirect Addressing Mode

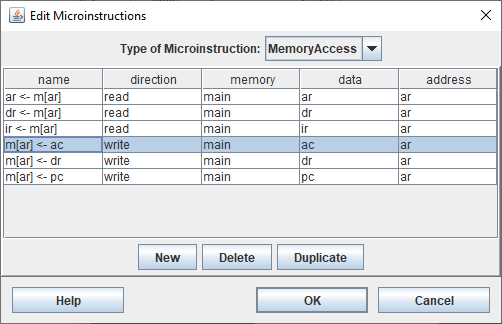


###### Implementation of STA\_I

* **AR <- M[AR]**



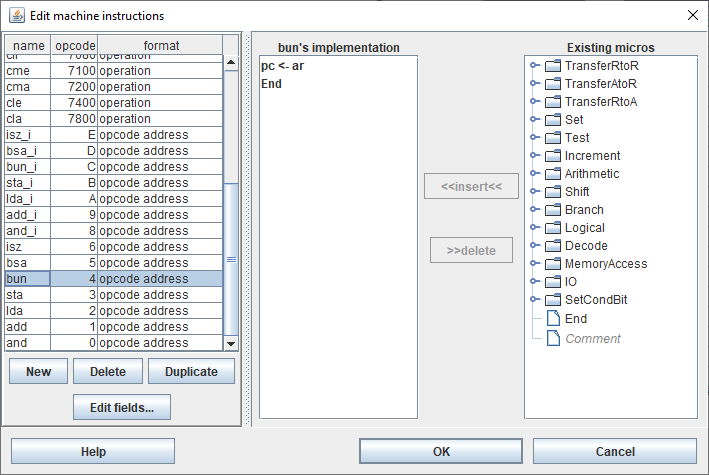
* **M[AR] <- AC**



* **SC <- 0**

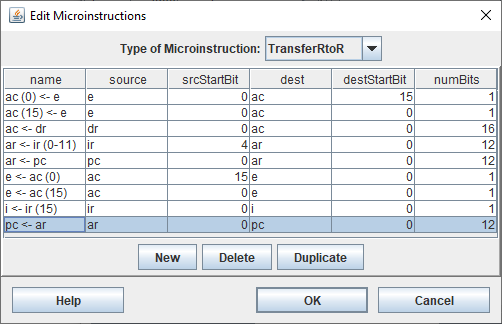
#### Branch UNconditionally

##### BUN – Direct Addressing Mode



###### Implementation of BUN

* **PC <- AR**



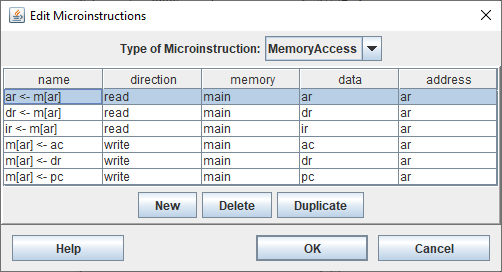
* **SC <- 0**

##### BUN\_I – Indirect Addressing Mode

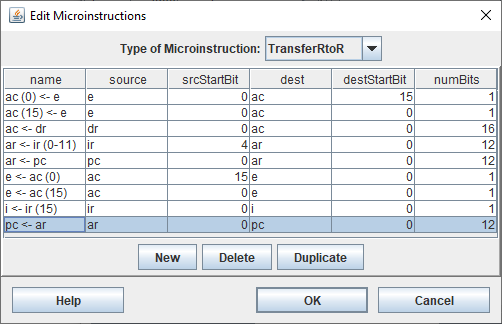
###### 

###### Implementation of BUN\_I

* **AR <- M[AR]**



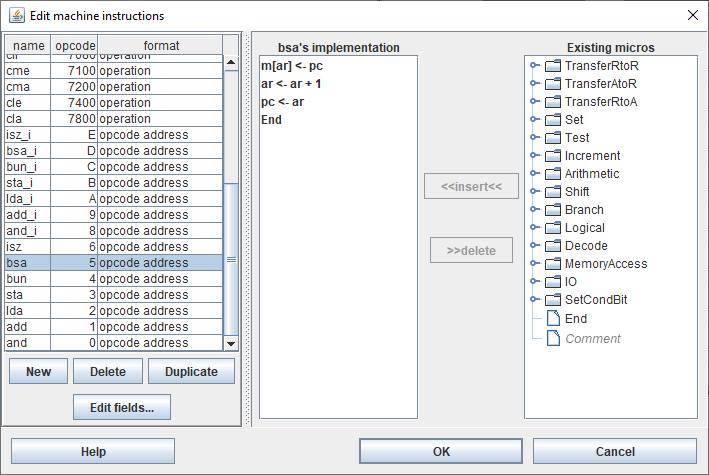
* **PC <- AR**



* **SC <- 0**

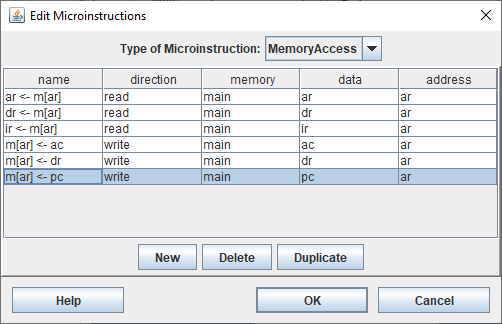
#### Branch and Save return Address

##### BSA – Direct Addressing Mode

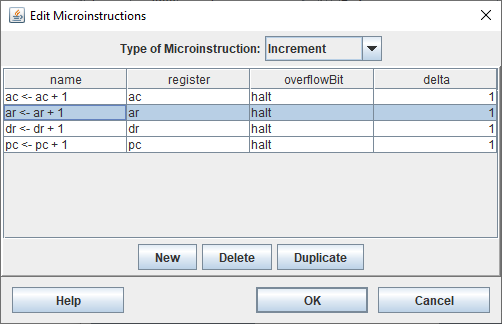


###### Implementation of BSA

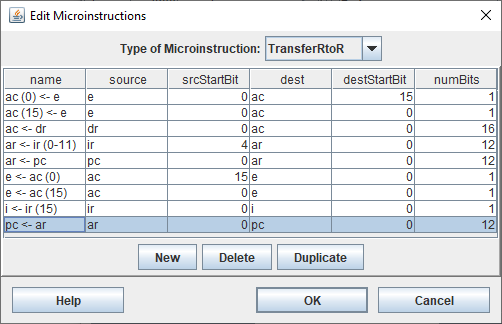
* **M[AR] <- PC**



* **AR <- AR + 1**

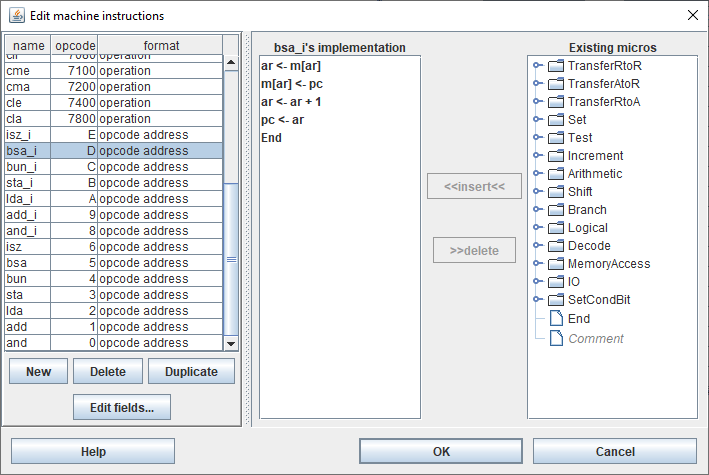


* **PC <- AR**



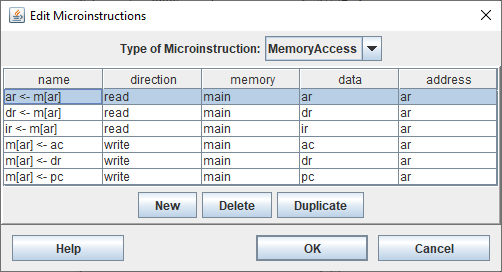
* **SC <- 0**

##### BSA\_I – Indirect Addressing Mode

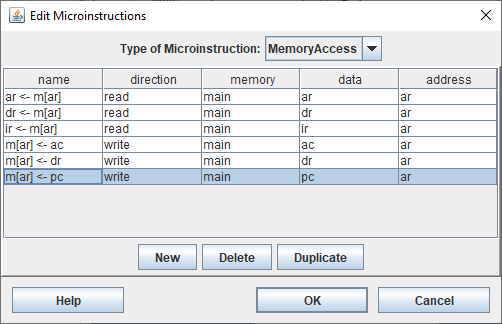


###### Implementation of BSA

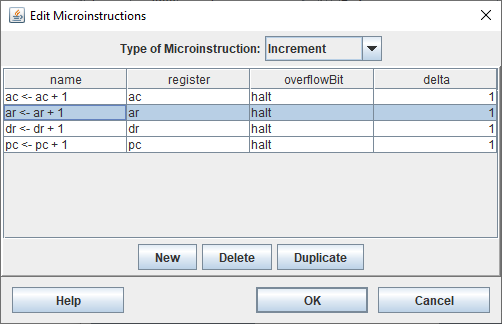
* **AR <- M[AR]**



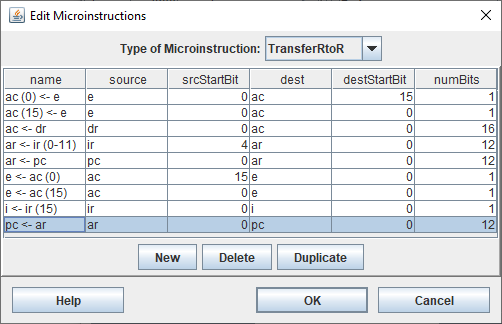
* **M[AR] <- PC**



* **AR <- AR + 1**



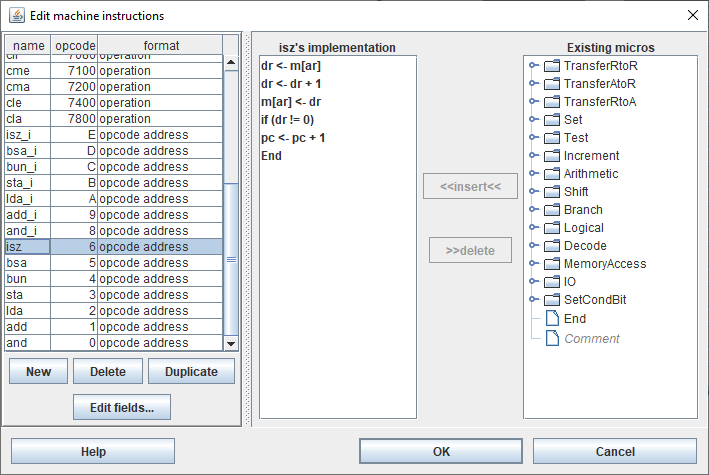
* **PC <- AR**



* **SC <- 0**

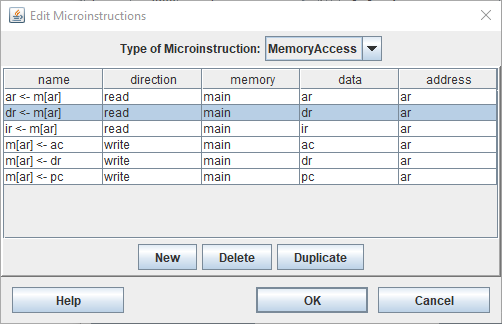
#### Increment and Skip If Zero

##### ISZ – Direct Addressing Mode

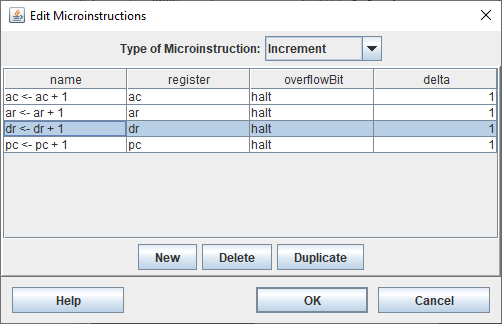


###### Implementation of ISZ

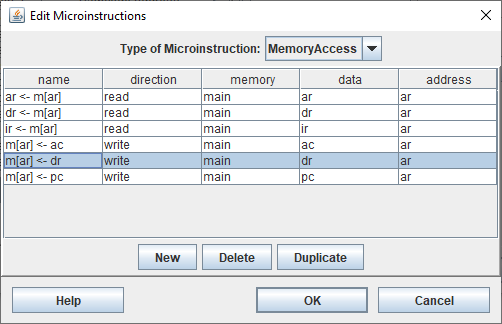
* **DR <- M[AR]**



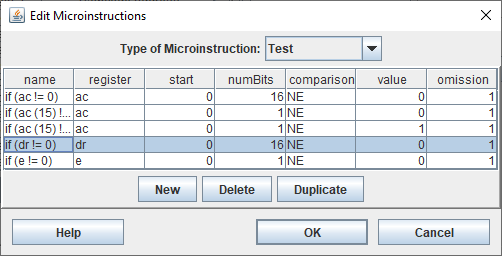
* **DR <- DR + 1**



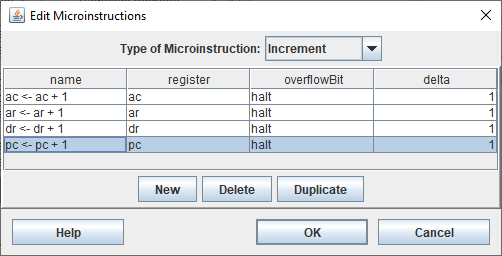
* **M[AR] <- DR**



* **If DR != 0** (CPUSim skips a micro-operation when this is true)

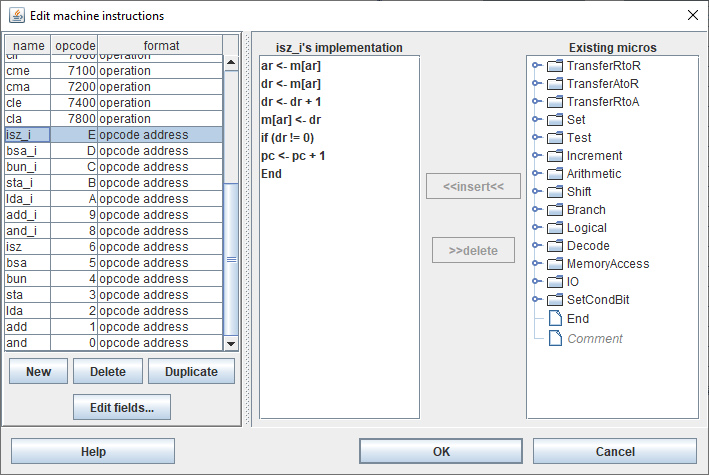


* **PC <- PC + 1**



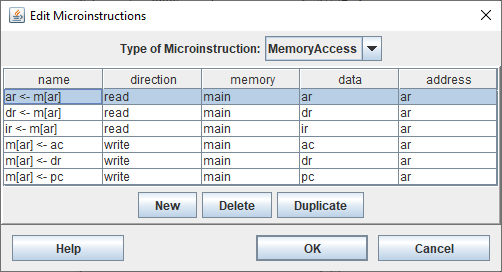
* **SC <- 0**

##### ISZ\_I – Indirect Addressing Mode

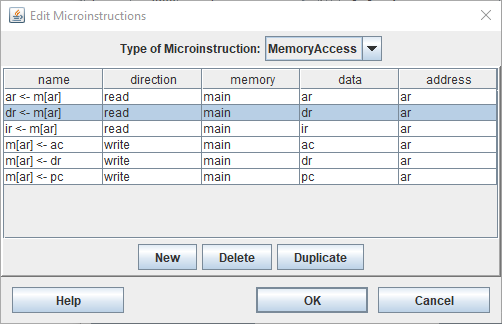


###### Implementation of ISZ\_I

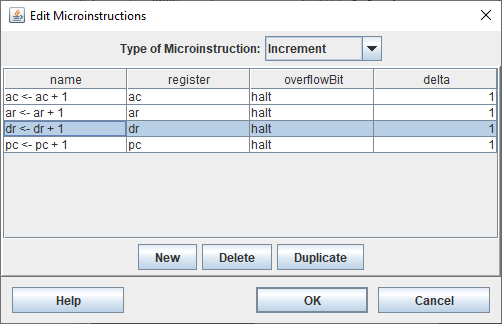
* **AR <- M[AR]**



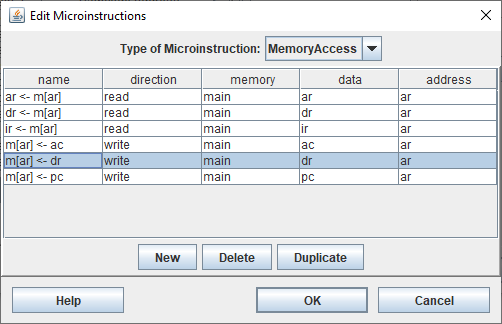
* **DR <- M[AR]**



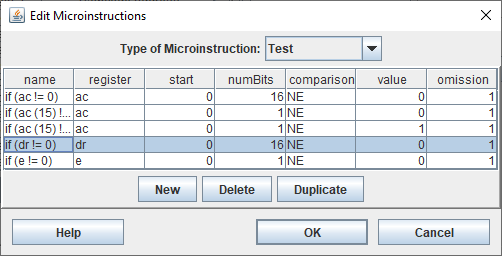
* **DR <- DR + 1**



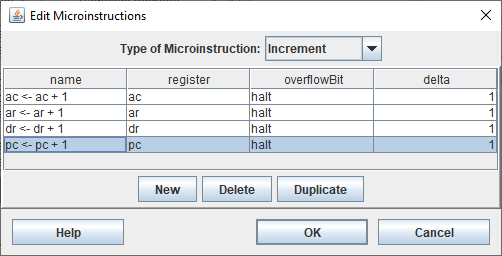
* **M[AR] <- DR**



* **If DR != 0** (CPUSim skips a micro-operation when this is true)



* **PC <- PC + 1**

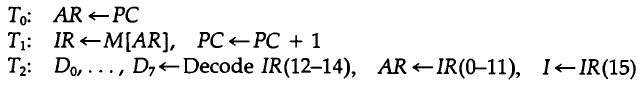


* **SC <- 0**

# PRACTICAL 3

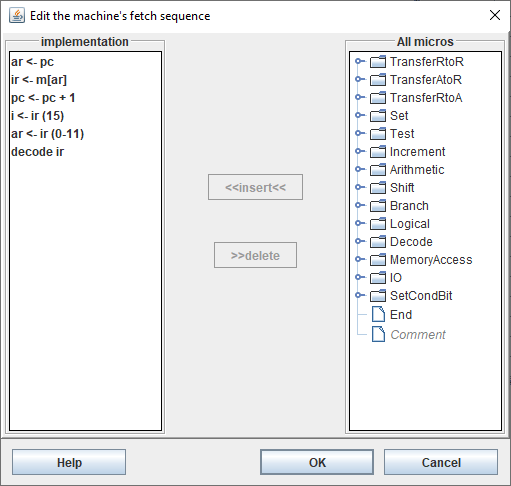
Dated 10th September 2019

## Objective

Create the fetch routine for the basic computer as designed in the previous practical.

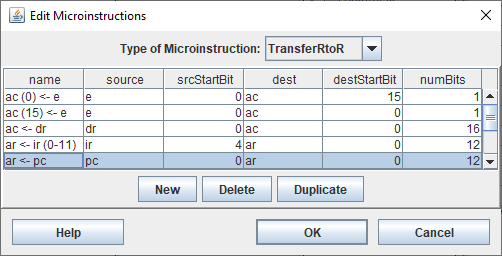
## Observation

### Fetch Sequence

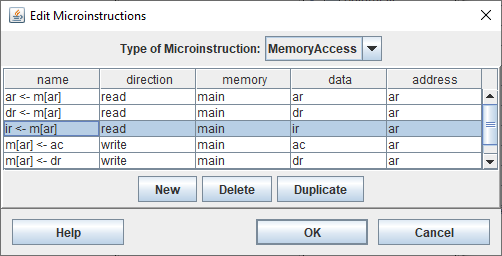


#### Implementation of the Fetch Sequence

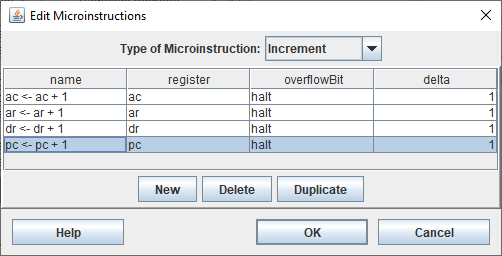
* **AR <- PC**



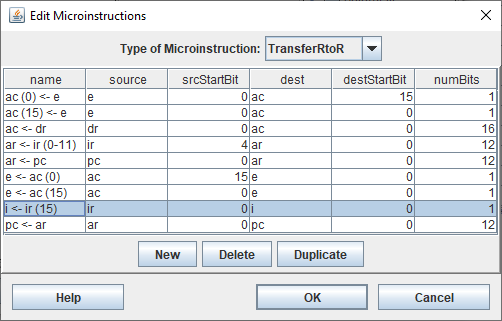
* **IR <- M[AR]**



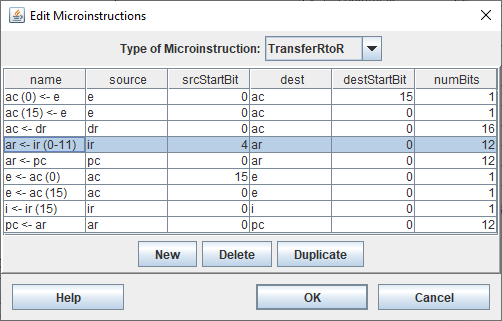
* **PC <- PC + 1**



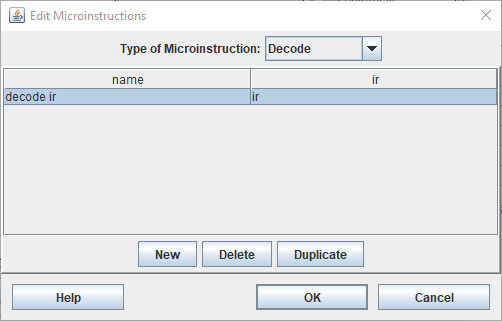
* **I <-IR (15)**



* **AR <- IR (0-11)**



* **Decode IR**



# PRACTICAL 4

Dated 24th September 2019

## Objective

|  |  |  |
| --- | --- | --- |
| 1. CLA | (e) CIR | (i) SNA |
| 1. CLE | (f) CIL | (j) SZA |
| 1. CMA | (g) INC | (k) SZE |
| 1. CME | (h) SPA | (l) HLT |

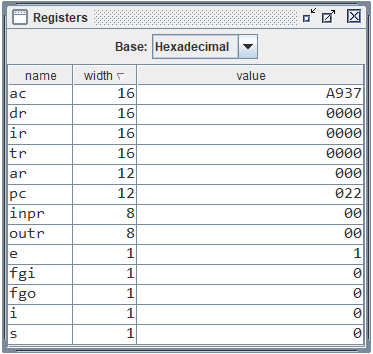
Simulate the machine to determine the contents of AC, E, PC, AR, and IR registers in hexadecimal after the execution of each of the following register reference instructions:

Initialize the contents of AC to (A937)16, that of PC to (022)16 and E to 1.

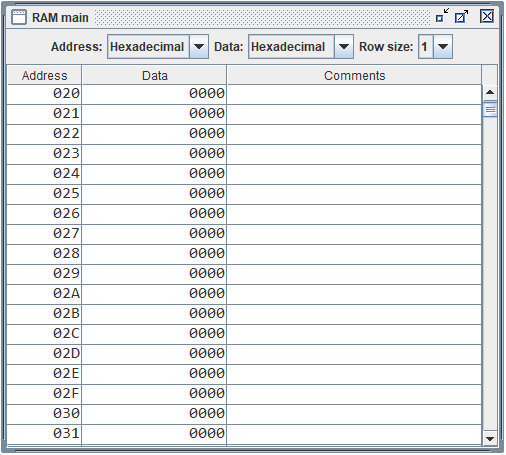
## Observation

### Before Loading

#### Registers



#### Memory

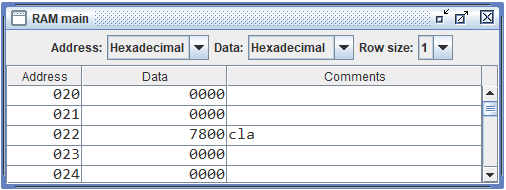


### CLA

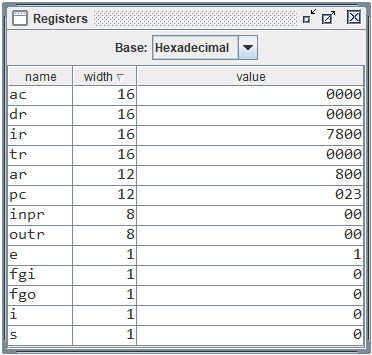
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

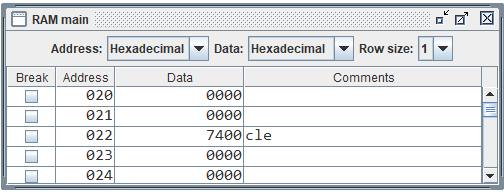


### CLE

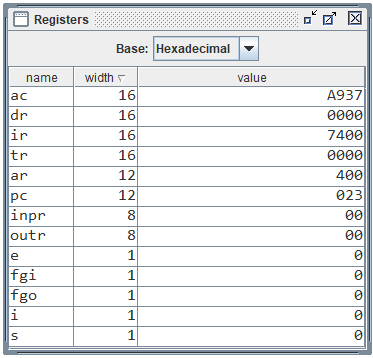
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

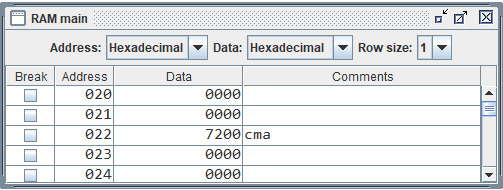


### CMA

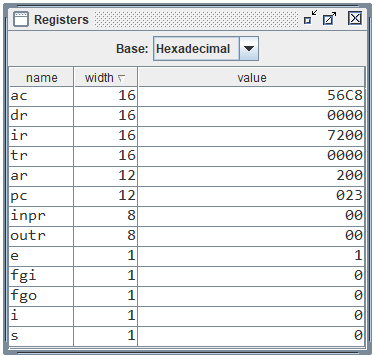
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

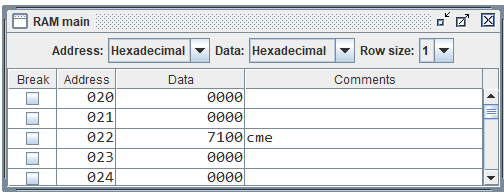


### CME

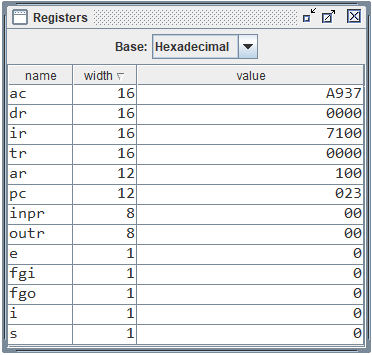
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

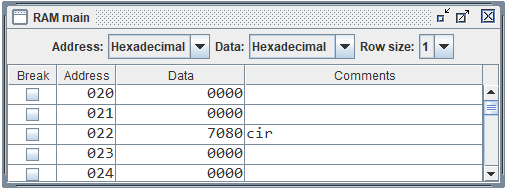


### CIR

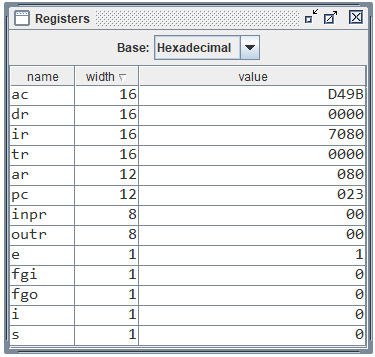
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

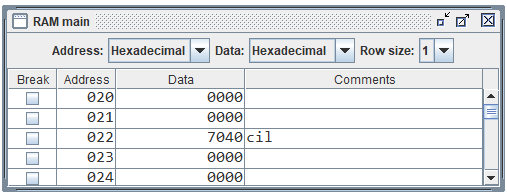


### CIL

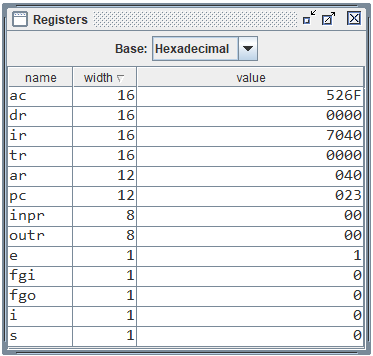
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

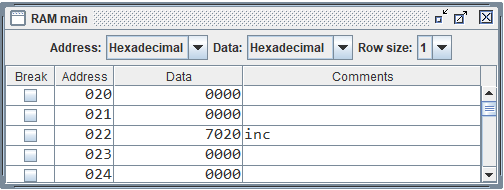


### INC

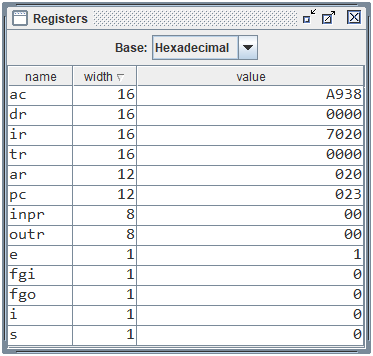
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

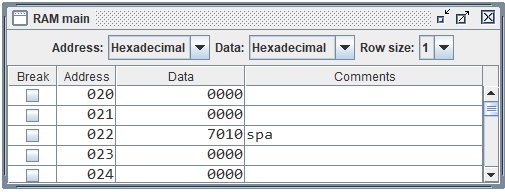


### SPA

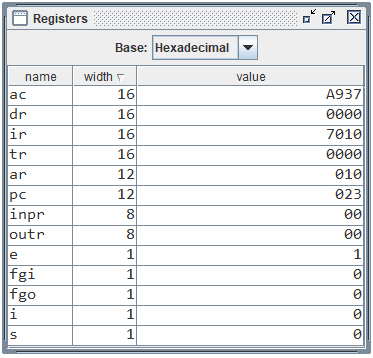
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

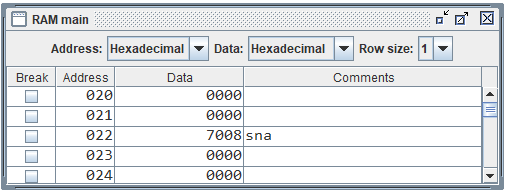


### SNA

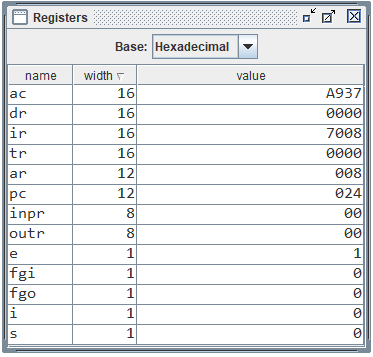
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

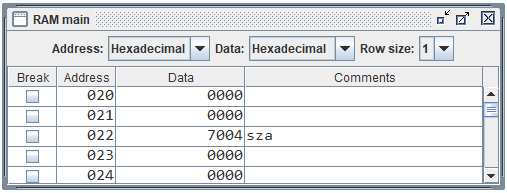


### SZA

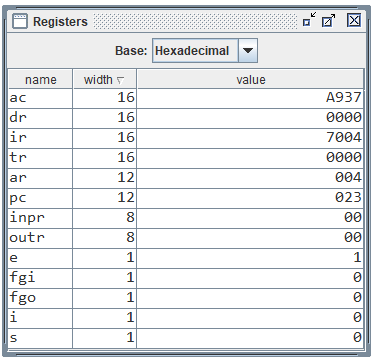
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

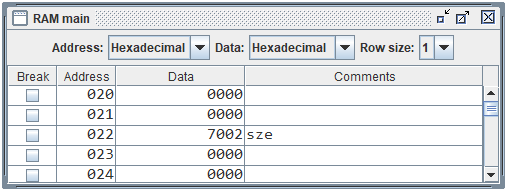


### SZE

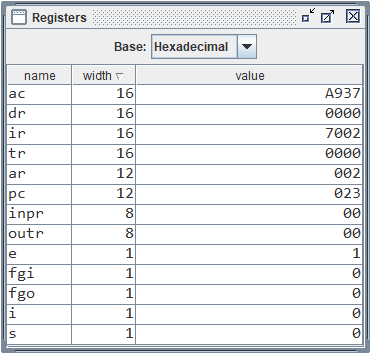
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

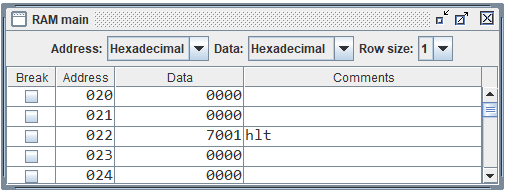


### HLT

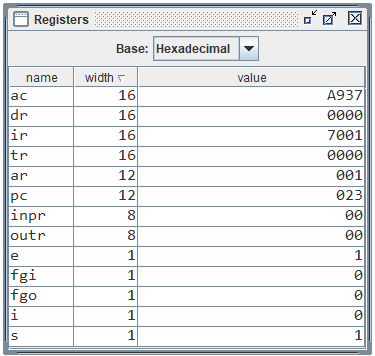
#### Assembly Program

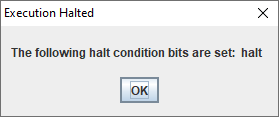


#### Memory After Loading



#### Registers After Execution





# PRACTICAL 5

Dated 24th September 2019

## Objective

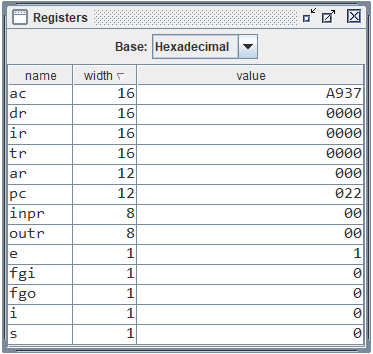
Simulate the machine for the following memory-reference instructions with I = 0 and address part = 082. The instruction is to be stored at address 022 in RAM. Initialize the memory word at address 082 with the operand B8F2 and AC with A937. Determine the contents of AC, E, PC, AR, and IR registers in hexadecimal after the execution.

|  |  |  |
| --- | --- | --- |
| 1. ADD | (d) STA | (g) ISZ |
| 1. AND | (e) BUN |
| 1. LDA | (f) BSA |

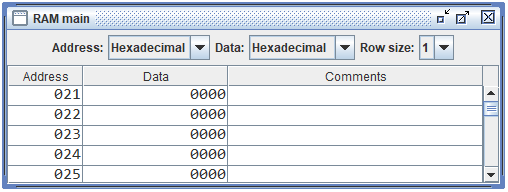
## Observation

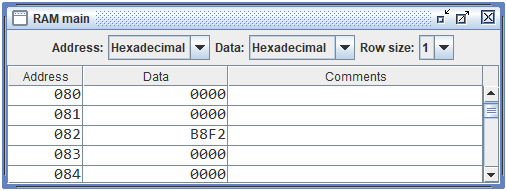
### Before Loading

#### Registers



#### Memory



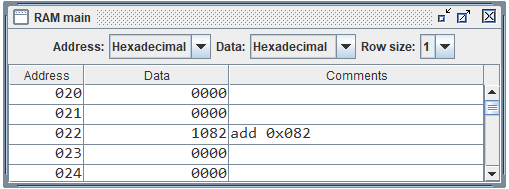


### ADD

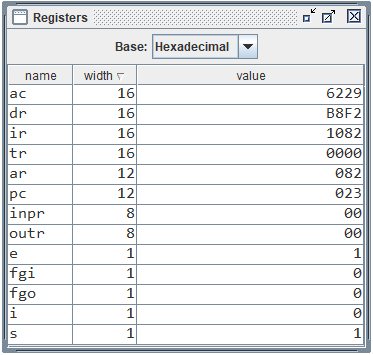
#### Assembly Program

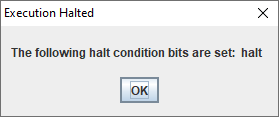


#### Memory After Loading



#### Registers After Execution



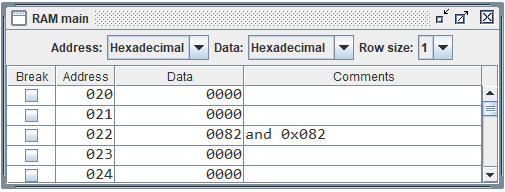


### AND

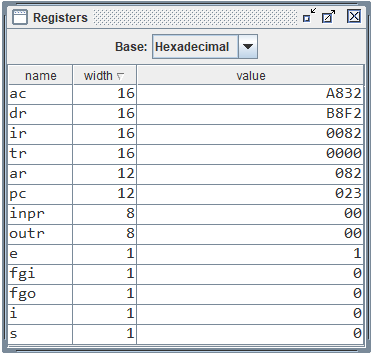
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

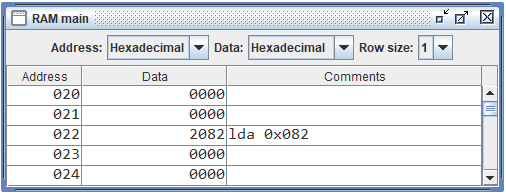


### LDA

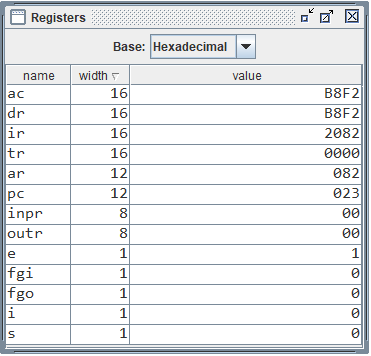
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

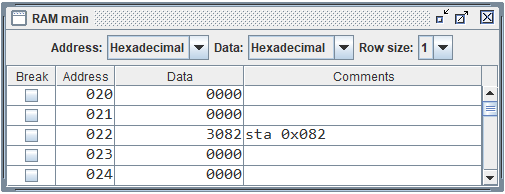


### STA

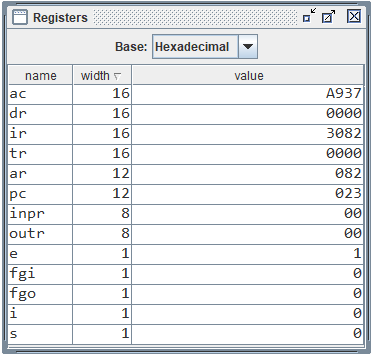
#### Assembly Program



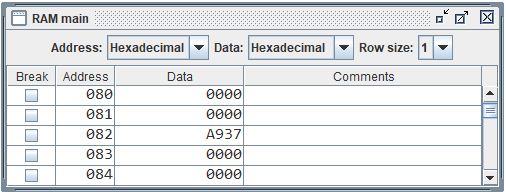
#### Memory After Loading



#### Registers After Execution



#### Memory After Execution

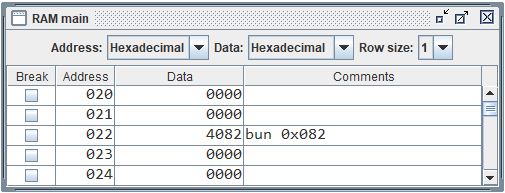


### BUN

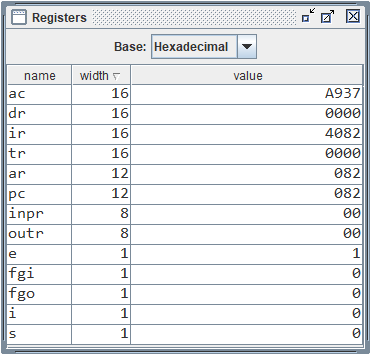
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

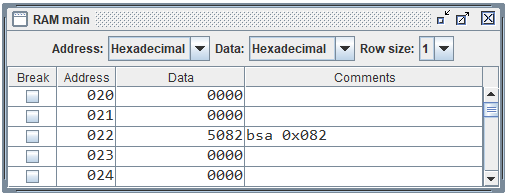


### BSA

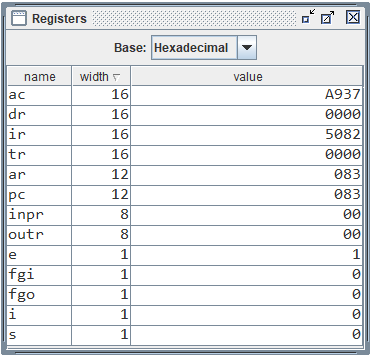
#### Assembly Program



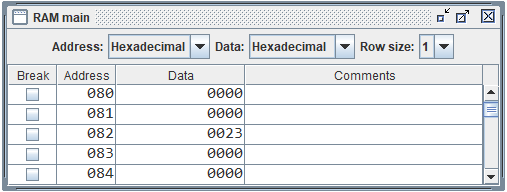
#### Memory After Loading



#### Registers After Execution



#### Memory After Execution

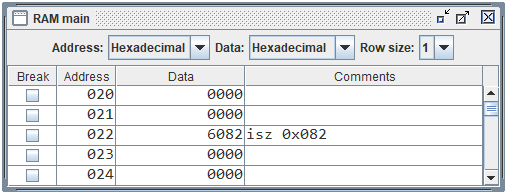


### ISZ

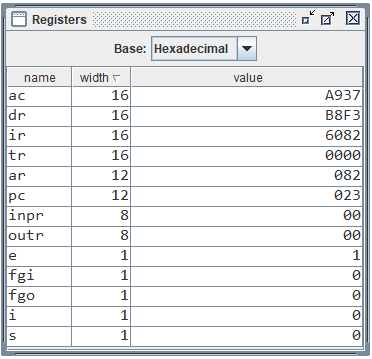
#### Assembly Program



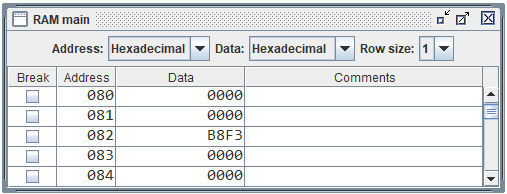
#### Memory After Loading



#### Registers After Execution



#### Memory After Execution



# PRACTICAL 6

Dated 24th September 2019

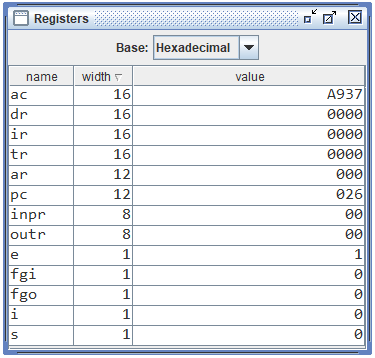
## Objective

Simulate the machine for the memory-reference instructions referred in above question with I = 1 and address part = 082. The instruction to be stored at address 026 in RAM. Initialize the memory word at address 082 with the value 298. Initialize the memory word at address 298 with operand B8F2 and AC with A937. Determine the contents of AC, DR, PC, AR and IR in hexadecimal after the execution.

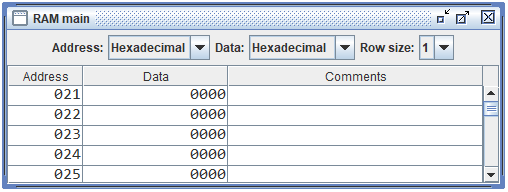
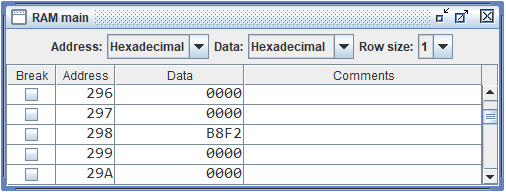
## Observation

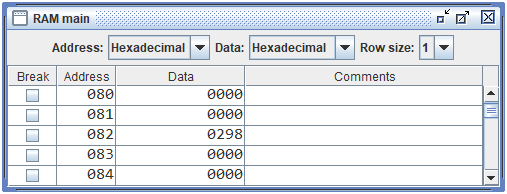
### Before Loading

#### Registers



#### Memory



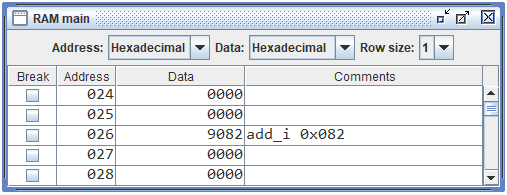


### ADD

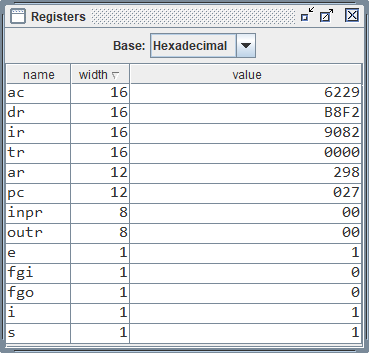
#### Assembly Program

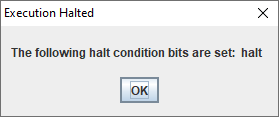


#### Memory After Loading



#### Registers After Execution



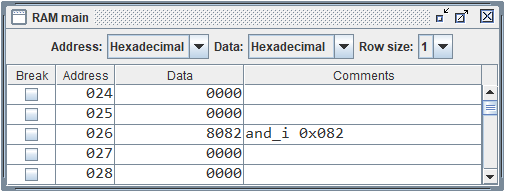


### AND

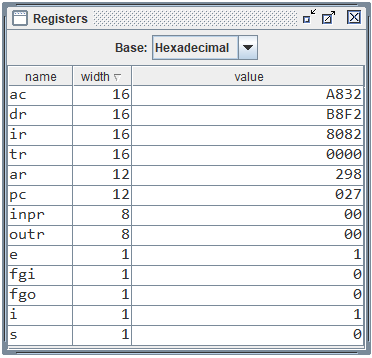
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

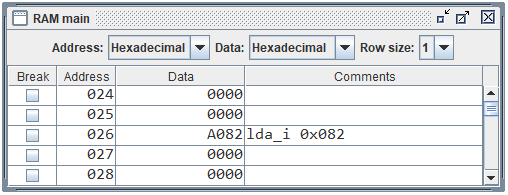


### LDA

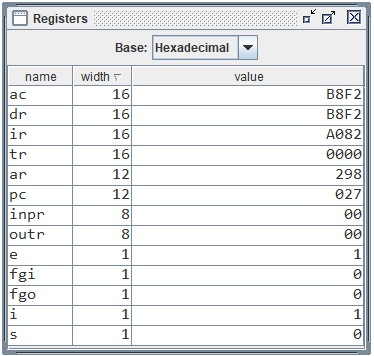
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

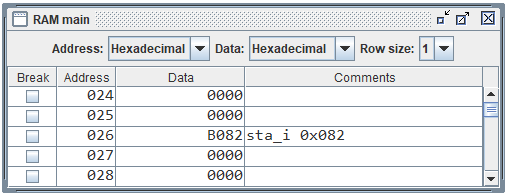


### STA

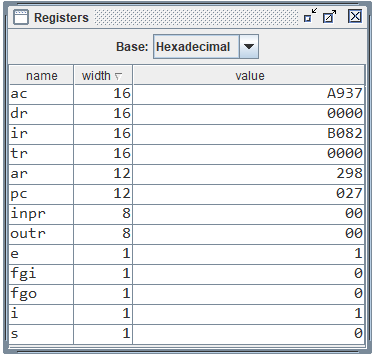
#### Assembly Program



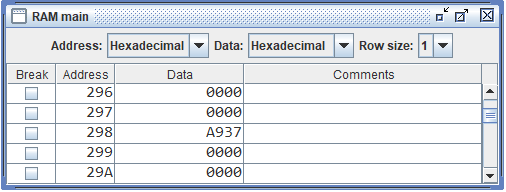
#### Memory After Loading



#### Registers After Execution



#### Memory After Execution

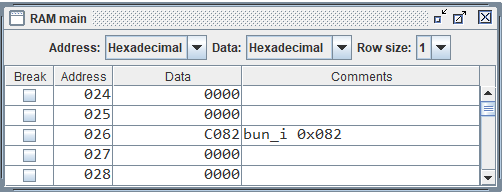


### BUN

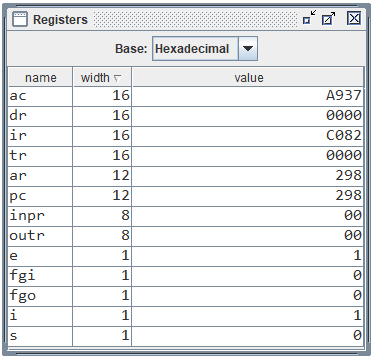
#### Assembly Program



#### Memory After Loading



#### Registers After Execution

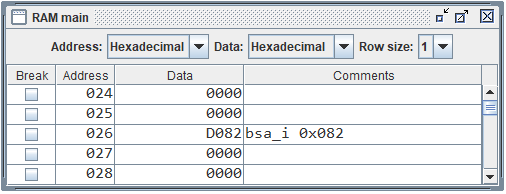


### BSA

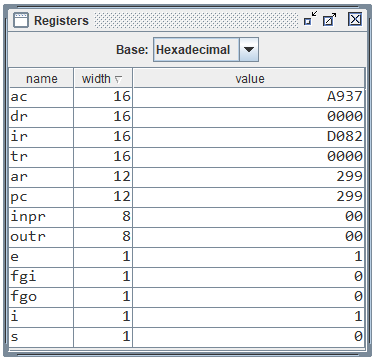
#### Assembly Program



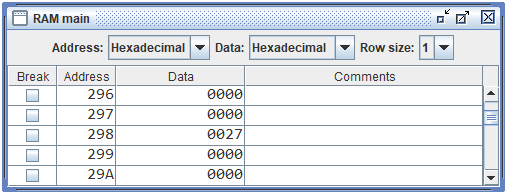
#### Memory After Loading



#### Registers After Execution



#### Memory After Execution

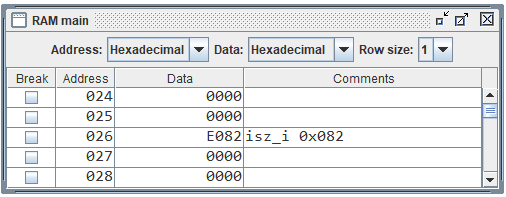


### ISZ

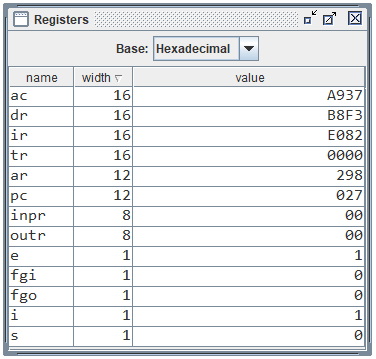
#### Assembly Program



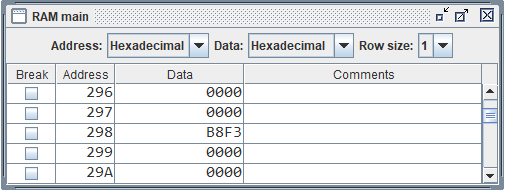
#### Memory After Loading



#### Registers After Execution



#### Memory After Execution

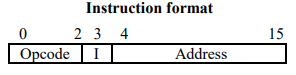


# PRACTICAL 7

Dated 01st October 2019

## Objective

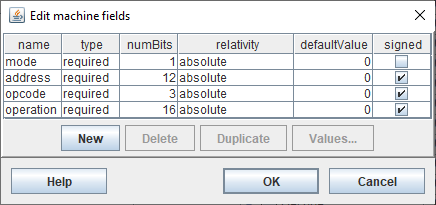
Modify the machine created in Practical 1 according to the following instruction format:



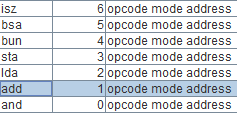
1. The instruction format contains a 3-bit opcode, a 1-bit addressing mode and a 12-bit address. There are only two addressing modes, I = 0 (direct addressing) and I = 1 (indirect addressing).
2. Create a new register I of 1 bit.
3. Create two new microinstructions as follows:
   1. Check the opcode of instruction to determine type of instruction (Memory-Reference/Register-Reference/Input-Output) and then jump accordingly.
   2. Check the I bit to determine the addressing mode and then jump accordingly.

## Observation

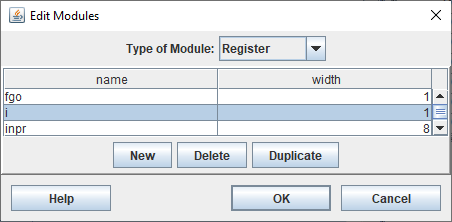
### Machine Fields



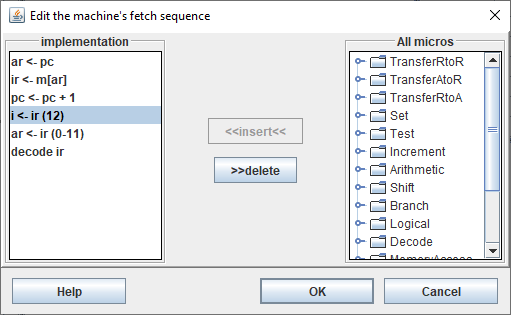
### Instruction Format

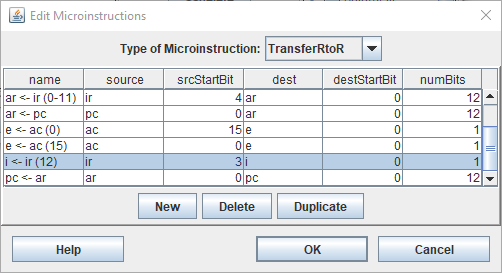


### Registers



### Fetch Sequence

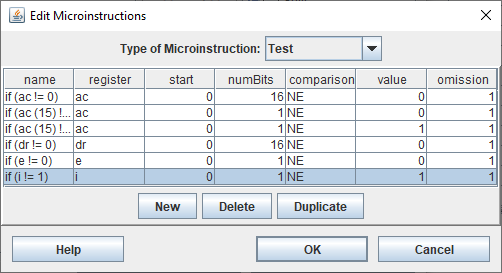




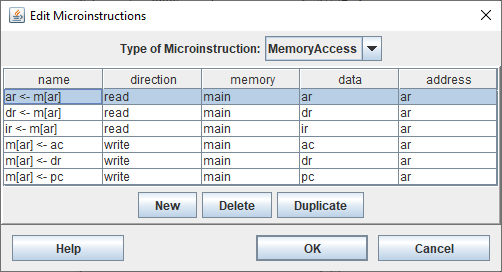
Extracting Mode Bit from IR

### Checking Addressing Mode

* **If (i != 1)** (CPUSim skips a micro-operation when this is true)

****

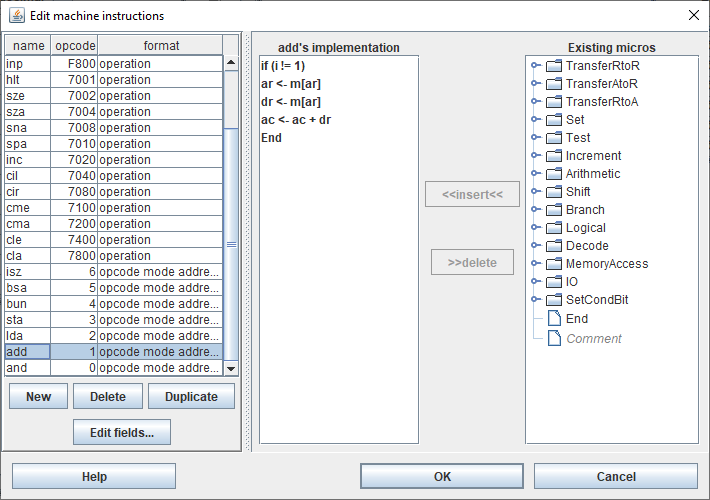
* **ar <- m[ar]**



* **other microinstructions follow**

### Example

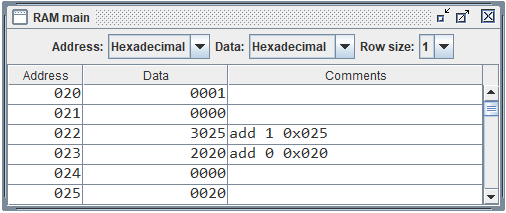
#### Implementation of ADD



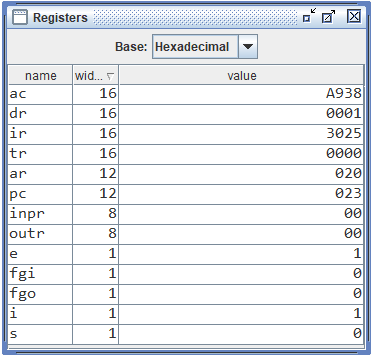
#### Assembly Code



#### Memory After Loading



#### Registers After Execution of Instruction at 0x022



#### Registers After Execution of Instruction at 0x023

