PROJECT 3

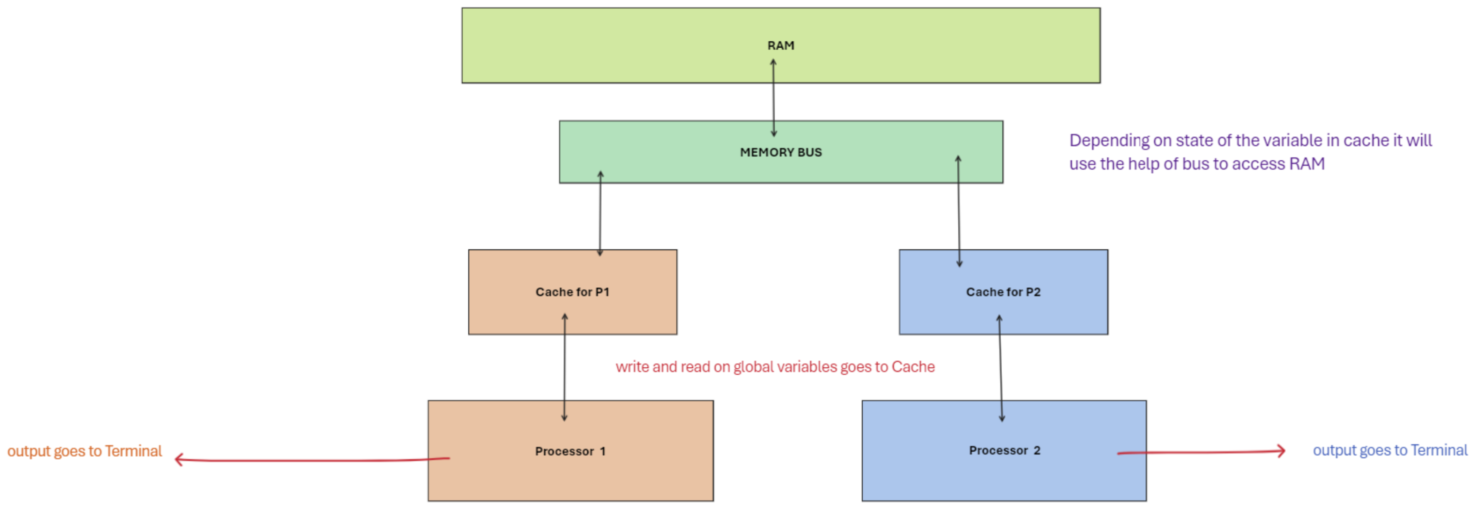
Sai vishnu Anudeep Kadiyala
001450445

NOVEMBER 30TH, 2023

TABLE OF CONTENTS:

System Documentation:

a High-level dataflow diagram.


RAM
MEMORY BUS
Depending on state of the variable in cache it will use the help of bus to access RAM
Cache for P1
Cache for P2
write and read on global variables goes to Cache
output goes to Terminal
Processor 1
Processor 2
output goes to Terminal

As mentioned in the above image this project is basically built on the project 3 implementation of Memory bus and States of each variable. So now we include Cache which I individual for both the processors and it basically is used to something that will hold up to 3 variables and if it is about to overflow it will replace the Least Reused Variable all the current 3 variables. The Cache now maintains the states instead of Symbol Table *[Variable Container in my case.]*

b List of Routines and descriptions:

So Initially addresstovaluedict.c have a new attribute added, where it will now every time when an insert happens on a variable name in map it will keep track of time, which is there after used to find out Least Reused Variable.

- <u>addresstovaluedict.c</u>

Function name:	Function usage:
InsertTimeStamp	InsertTimeStamp: takes map and key in the map and if the key is found we update the timetakes map and key in the map and if the key is found we update the timev
InsertBasingOnIndex	so, this is for cache memeory specially as in cache we have to replace the least reused item with the new item, so we use this to repalce the index of least reused with the new item.
atIndexInMap	at the provided index we return the attributes we have in that index.
compareTimes	used to compare two timespec times of two variables to findout which one is the oldes -- least reasused component time comparision function
leastRecentlyUsedIndex	from the given map we search for the oldes lastUsedTime and return that index to replace next.

- <u>CacheMemory.c</u>

Function name	Function description
initializeCache	initialize the cache, with size of 3
InsertInCache	This is the method that is responsible for inserting new elements at right spots in cache This at step 1 looks at the cache and checks if it is already available in cache, if it is available then it will just reinsert with changes and updates the lastUsedTime for this variable. while it reedits this, it will also for synced variables to resolve their values, In Step 2 if step 1 fails, it will no check if the size of cache < 3. if so then it will go ahead and insert the new element. else it will now call the IndextoUpdate method to findout the index with LRU and replace it using the new values and creates a new time stamp of lastUsedTime while this is the case if the element leaving cache is in M it will send to bus
FindInCache	finds in the Cache and returns the value of variable

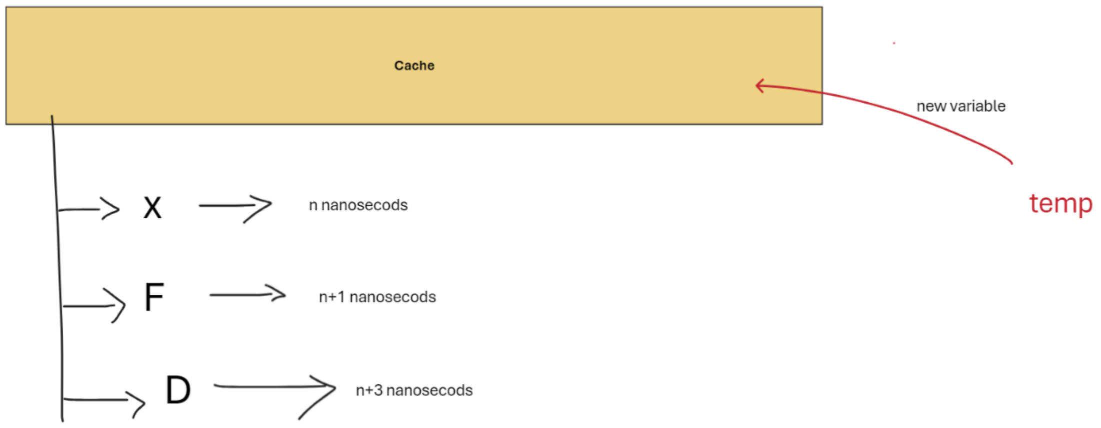
FindInCache_Status	find status of the given varname in cache.
FindInCache_SyncedWith	returns the index it is synced with
updateInCacheState	used to update the state of a varname in cache,
updateInCacheTimeStamp	updates the lastUsedTime of varname.
InsertSyncedAddress_InCache	used to update the synced of a variable when :& happens on symbol table.

These functions are used to wrap the cache memory functionalities and work around the story of Implementing Cache in between Memory bus and processor.

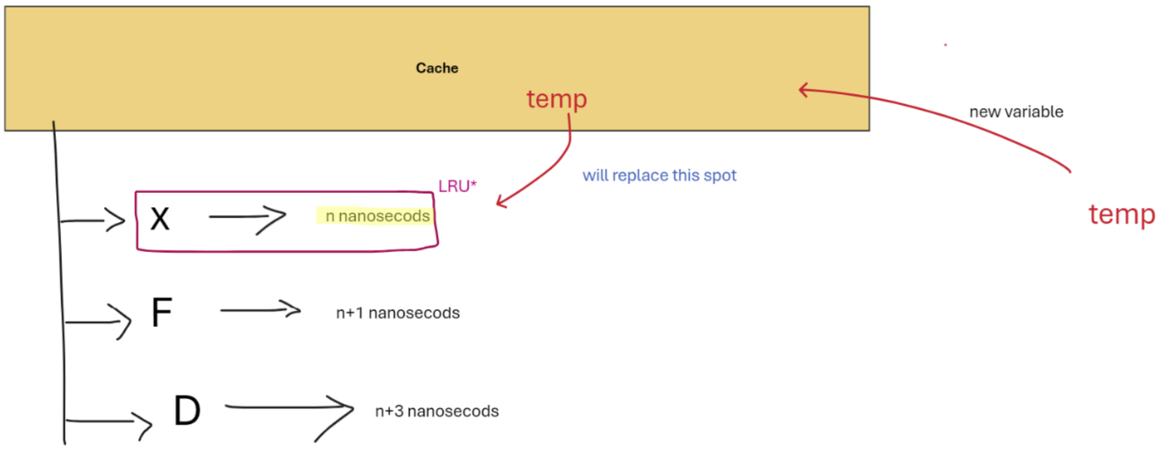
c <u>Implementation Method:</u>

Cache memory is responsible for handling read and write on Global variables, and it will load them on the space of 3 variables and accordingly if its filled, we replace the Least reused values, so the way I solved this case by using Time factor, the older the last used time of a variable is that is the least reused variable.

For example, let's take a case of cache where at that moment we have 3 variables in Cache, and we want to add 4th variable.



Now we see that temp is trying to enter Cache and when we see the times of variables, X is oldest with n nano seconds and all other time are later than this, hence we replace temp in place of X in Cache.



So, in such case, when a variable is leaving Cache and if it is in state M we will send the bus a signal to update the value as it is leaving Cache.

While this is the general explanation of the Cache and its LRU concept, there is another layer to this, how do we manage to write and read of the Cache.

So, when we read a variable value as a part of rvalue in abm code.
We have multiple cases that we need to address:
There are two primary cases.

1. **In Cache:**
   Now this has two more cases,
   a. **The given Varname is in I:** we read from Bus and update the value.
   b. **The given varname is in S/M:** we read from Cache directly and move forward in code,

2. **Not in Cache:**
   here we are supposed to read from BUS anyway as we don't have the value.
   Now after reading when we insert in Cache, we have two cases.
   a. **the variable is previously synced with other variable**: insert the variable in cache with synced address.
   b. **the variable is standalone**: insert the variable only.

Now moving towards writing data, this gets complicated as we need to account for that, even if this does have cases we need to account.
So, if the variable address we want to write in on Global Vars then now we have two possible outer cases.

1. **It is in cache:**
   Now if it is in cache this will have two other inner cases, after sending a bus signal if the given Varname is in state S or I in cache.

   a. The varname has a synced address:
   if the varname have a synced address have the final layer of two cases,

   i. **The synced address is also in Cache:** Just Insert into the cache as the InsertInCache method will auto handle this,
   j. **The synced address is not in Cache:** not technically possible but still handled.

   b. **The varname dont have a synced address:** just insert the new value with state M for the given variable name.

2. **It is not in cache:**
   Now if it is not in Cache, we will of course send to bus that there is a write but then we check if there is any synced variable for this. And accordingly, we handle it by Inserting the Value.

Test Documentation:

a How Did I Test My Program:

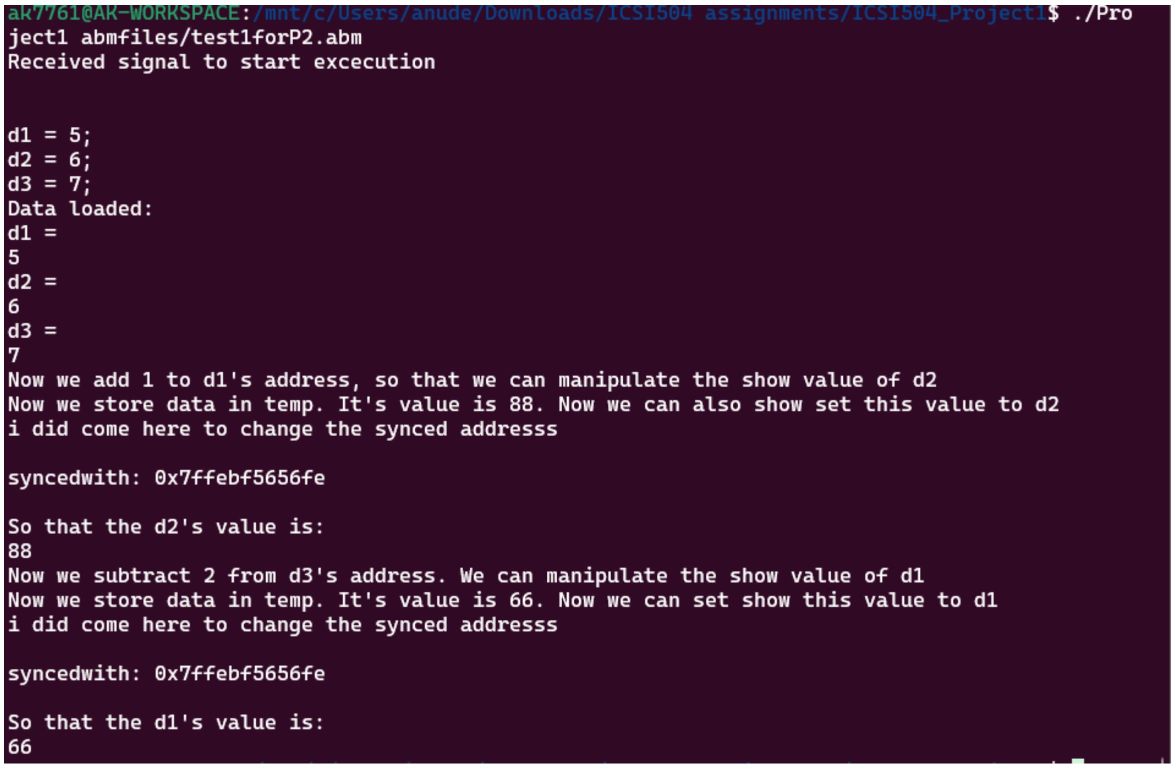
I have used given Testforp2.abm then in the submission abmfiles/pairedtest/ will have all the paired test cases that I used to validate my algorithms efficiency.

I have tested this code for address assignment accuracy, concurrency, and given Testforp2.abm also.

Apart from the requirements, I have written my own testcase pair, which is provided in abmfiles/pairedtest/Project3 where the pair uses more than 6 variables that have writes and reads to check the cache work. It is supposed to see 6 on both the terminals which I have achieved.

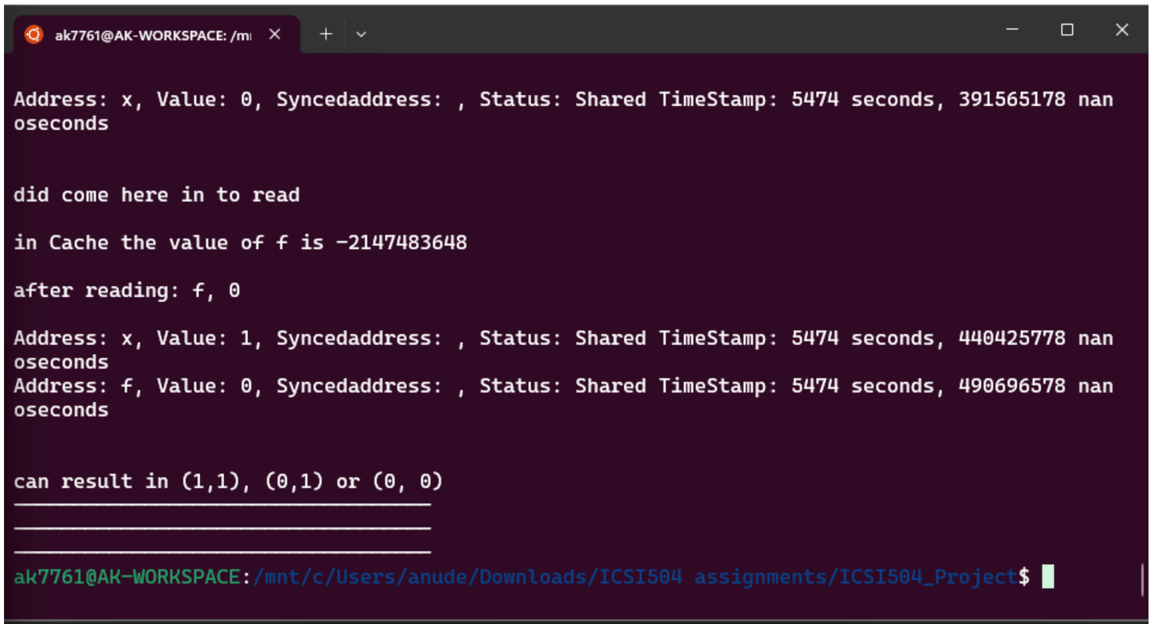
Below are the screenshots of outputs for each file. To mention they have matched exactly with each of the .out files provided showing the provided solutions credibility.

i Testforp2.abm:

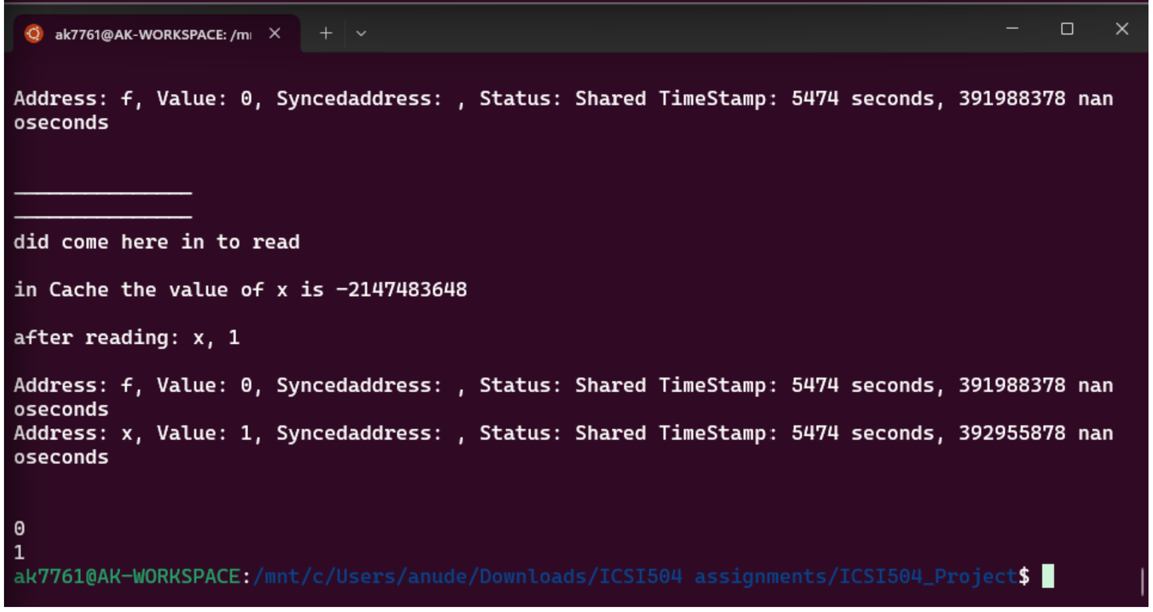


```
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/test1forP2.abm
Received signal to start excecution


d1 = 5;
d2 = 6;
d3 = 7;
Data loaded:
d1 =
5
d2 =
6
d3 =
7
Now we add 1 to d1's address, so that we can manipulate the show value of d2
Now we store data in temp. It's value is 88. Now we can also show set this value to d2
i did come here to change the synced addresss

syncedwith: 0x7ffebf5656fe

So that the d2's value is:
88
Now we subtract 2 from d3's address. We can manipulate the show value of d1
Now we store data in temp. It's value is 66. Now we can set show this value to d1
i did come here to change the synced addresss

syncedwith: 0x7ffebf5656fe

So that the d1's value is:
66
```

i *Pairedcases/Datarace Case:*

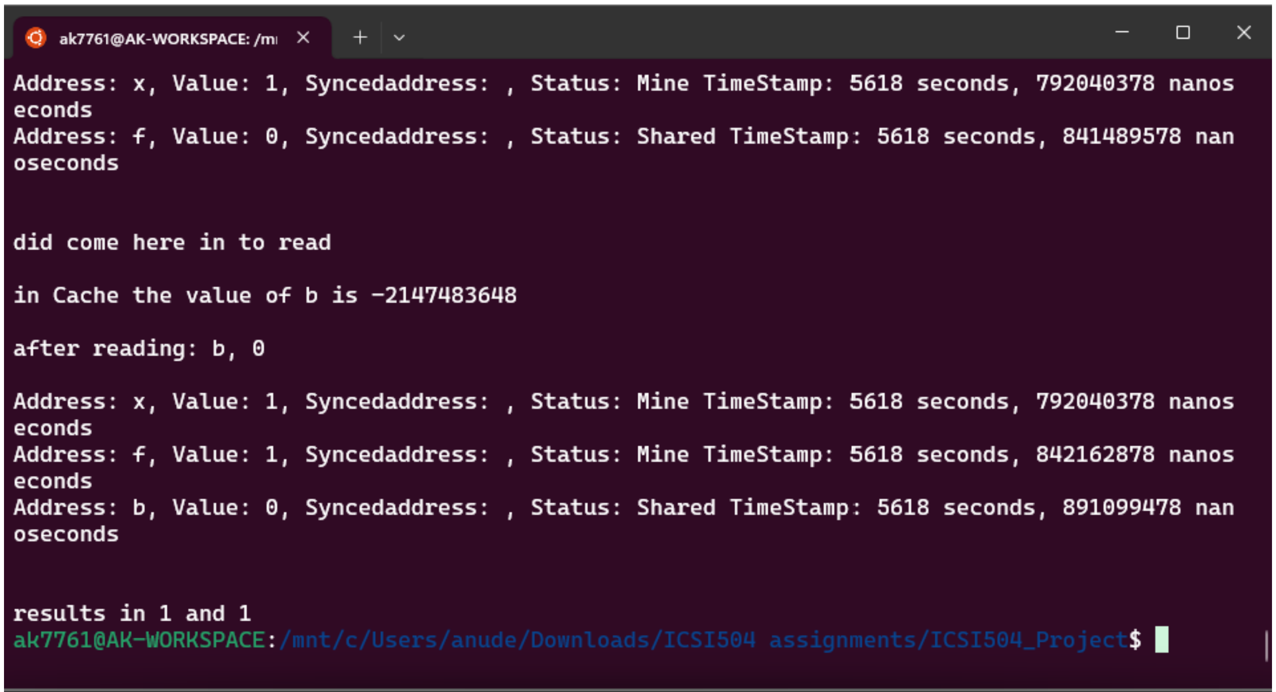


```
Address: x, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5474 seconds, 391565178 nan
oseconds


did come here in to read

in Cache the value of f is -2147483648

after reading: f, 0

Address: x, Value: 1, Syncedaddress: , Status: Shared TimeStamp: 5474 seconds, 440425778 nan
oseconds
Address: f, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5474 seconds, 490696578 nan
oseconds


can result in (1,1), (0,1) or (0, 0)
___________________________________
___________________________________
___________________________________

ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project$
```

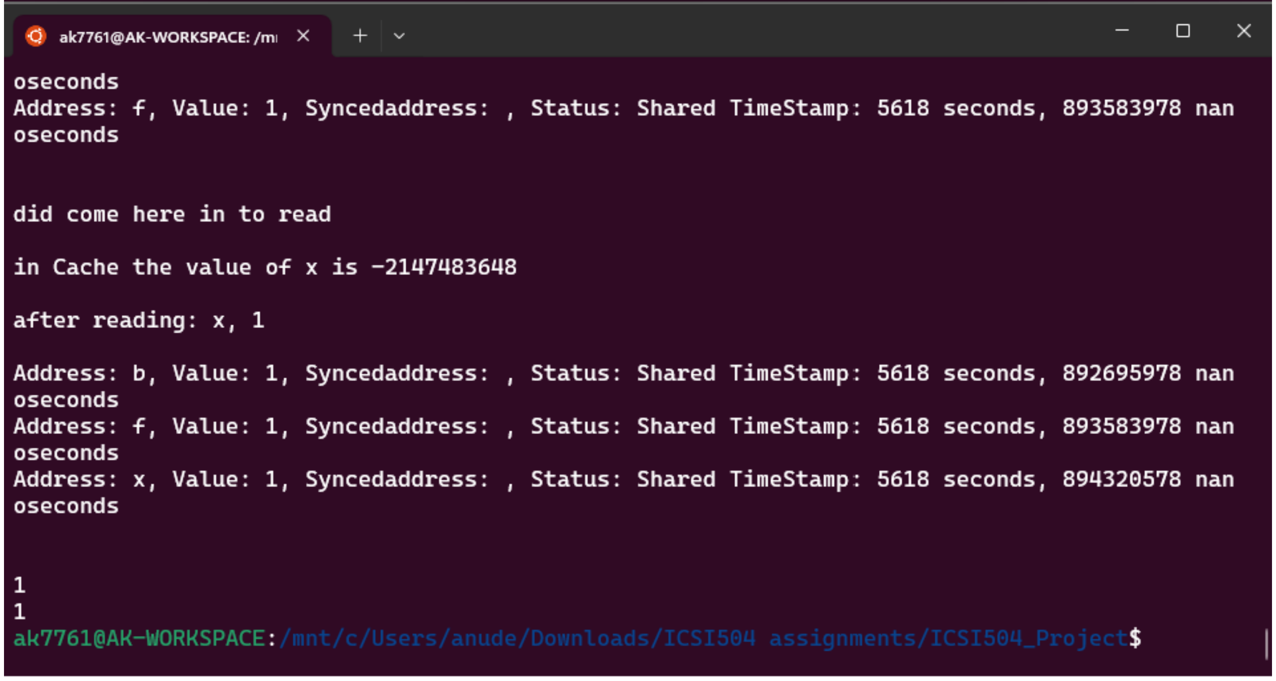


```
Address: f, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5474 seconds, 391988378 nan
oseconds


_______________
_______________
did come here in to read

in Cache the value of x is -2147483648

after reading: x, 1

Address: f, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5474 seconds, 391988378 nan
oseconds
Address: x, Value: 1, Syncedaddress: , Status: Shared TimeStamp: 5474 seconds, 392955878 nan
oseconds


0
1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project$
```

i *Bothof them are 1 case:*

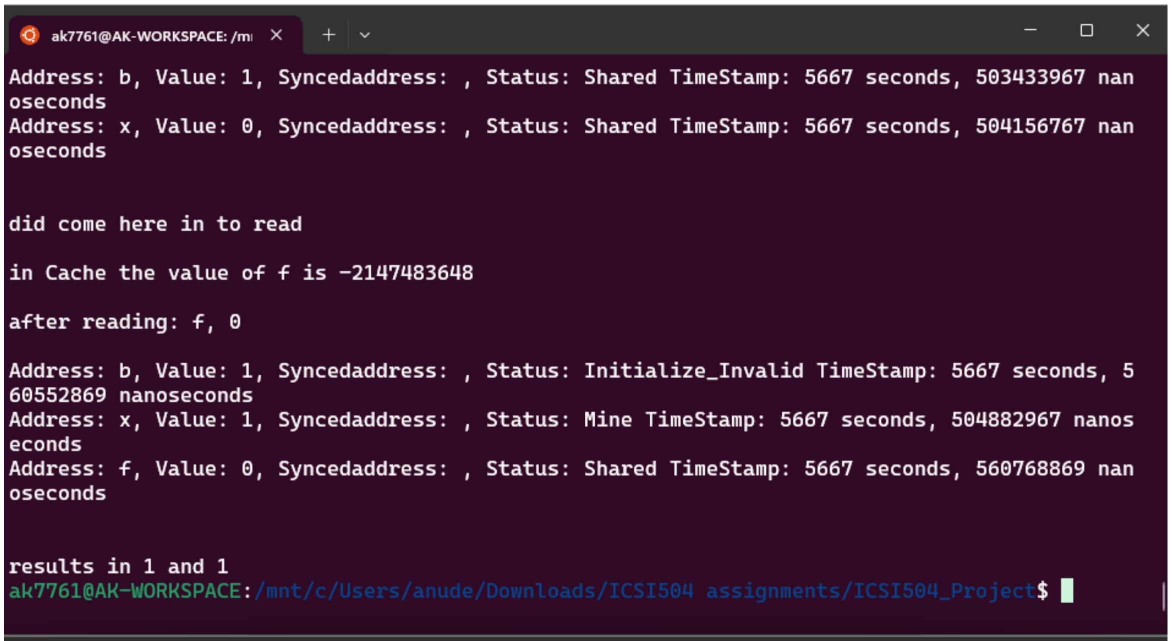


```
Address: x, Value: 1, Syncedaddress: , Status: Mine TimeStamp: 5618 seconds, 792040378 nanos
econds
Address: f, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5618 seconds, 841489578 nan
oseconds


did come here in to read

in Cache the value of b is -2147483648

after reading: b, 0

Address: x, Value: 1, Syncedaddress: , Status: Mine TimeStamp: 5618 seconds, 792040378 nanos
econds
Address: f, Value: 1, Syncedaddress: , Status: Mine TimeStamp: 5618 seconds, 842162878 nanos
econds
Address: b, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5618 seconds, 891099478 nan
oseconds


results in 1 and 1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project$
```

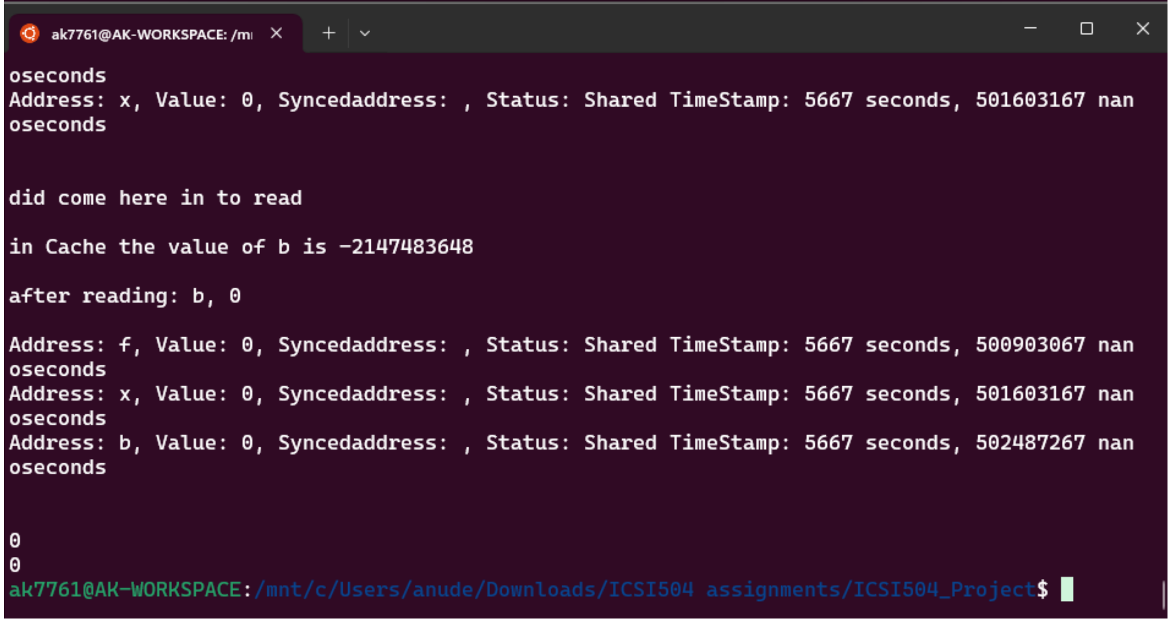


```
oseconds
Address: f, Value: 1, Syncedaddress: , Status: Shared TimeStamp: 5618 seconds, 893583978 nan
oseconds


did come here in to read

in Cache the value of x is -2147483648

after reading: x, 1

Address: b, Value: 1, Syncedaddress: , Status: Shared TimeStamp: 5618 seconds, 892695978 nan
oseconds
Address: f, Value: 1, Syncedaddress: , Status: Shared TimeStamp: 5618 seconds, 893583978 nan
oseconds
Address: x, Value: 1, Syncedaddress: , Status: Shared TimeStamp: 5618 seconds, 894320578 nan
oseconds


1
1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project$
```

iv *Both are 0 case :*

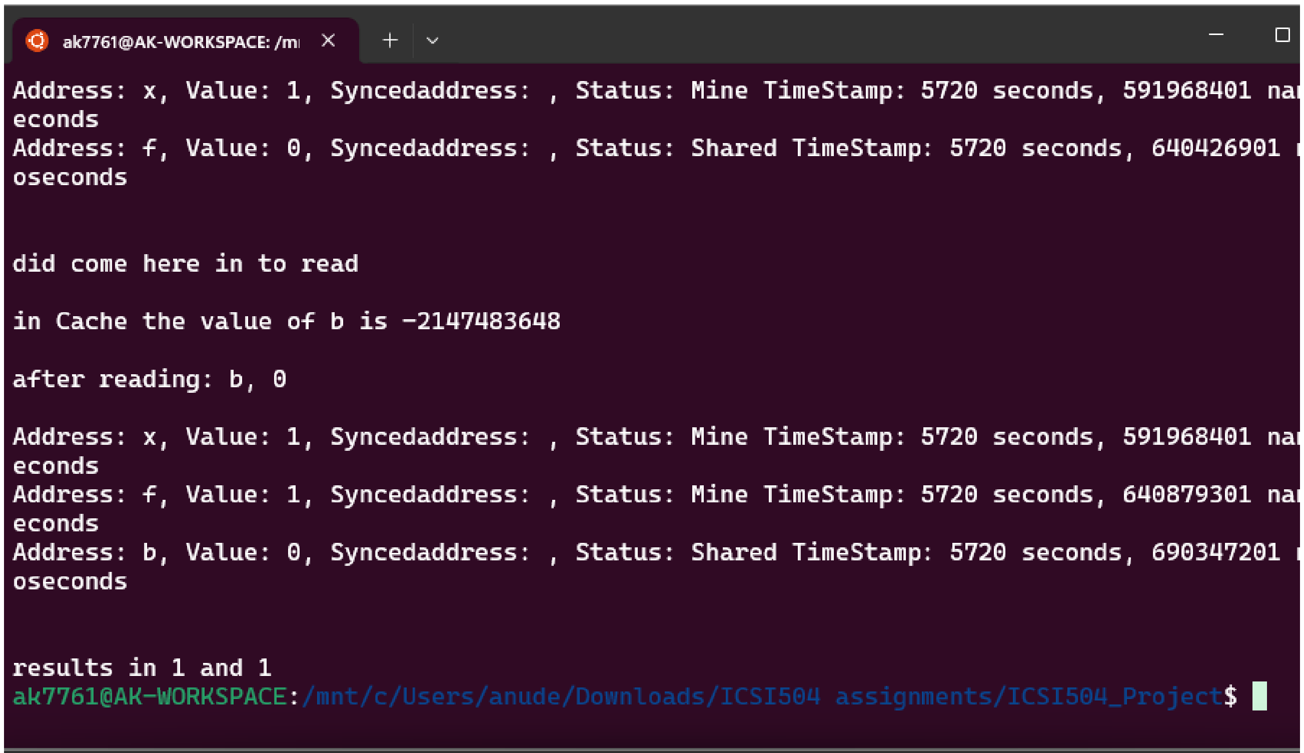


```
Address: b, Value: 1, Syncedaddress: , Status: Shared TimeStamp: 5667 seconds, 503433967 nan
oseconds
Address: x, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5667 seconds, 504156767 nan
oseconds


did come here in to read

in Cache the value of f is -2147483648

after reading: f, 0

Address: b, Value: 1, Syncedaddress: , Status: Initialize_Invalid TimeStamp: 5667 seconds, 5
60552869 nanoseconds
Address: x, Value: 1, Syncedaddress: , Status: Mine TimeStamp: 5667 seconds, 504882967 nanos
econds
Address: f, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5667 seconds, 560768869 nan
oseconds


results in 1 and 1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project$
```

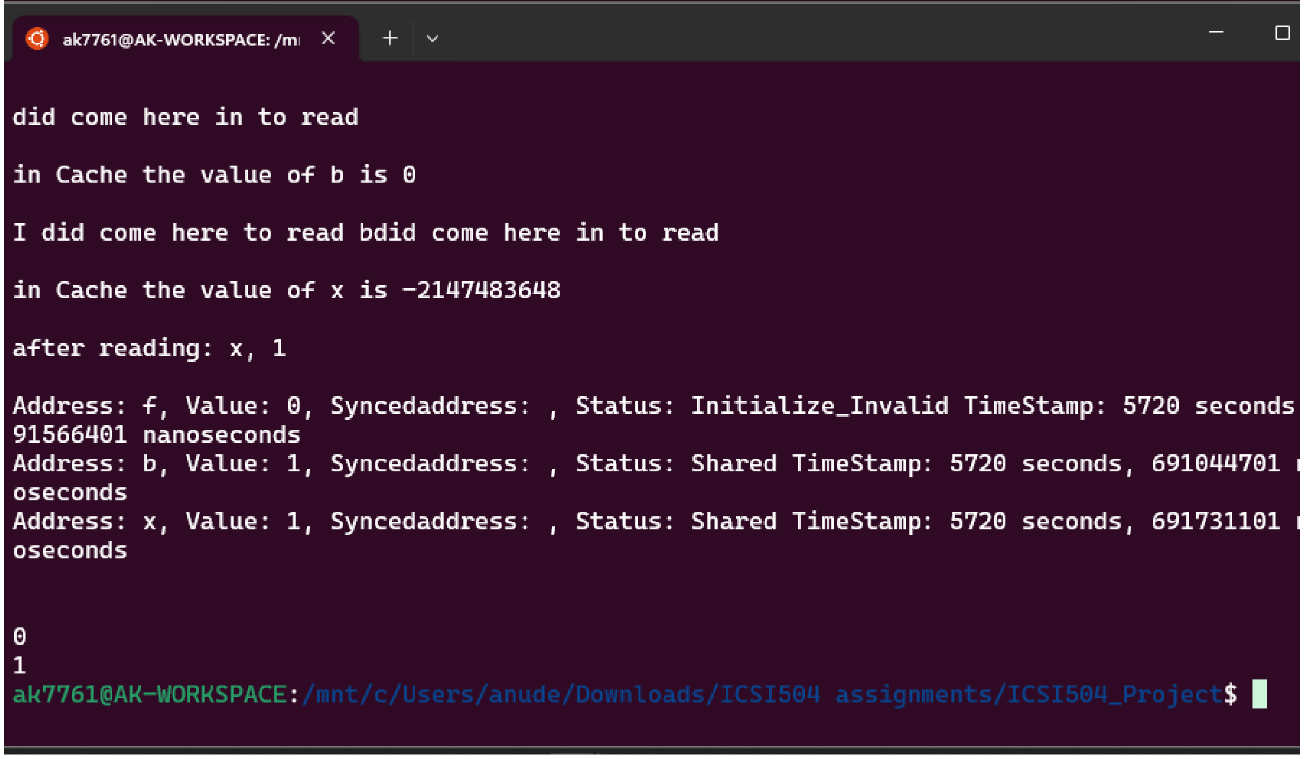


```
oseconds
Address: x, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5667 seconds, 501603167 nan
oseconds


did come here in to read

in Cache the value of b is -2147483648

after reading: b, 0

Address: f, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5667 seconds, 500903067 nan
oseconds
Address: x, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5667 seconds, 501603167 nan
oseconds
Address: b, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5667 seconds, 502487267 nan
oseconds


0
0
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project$
```

- *One of them Is 1:*

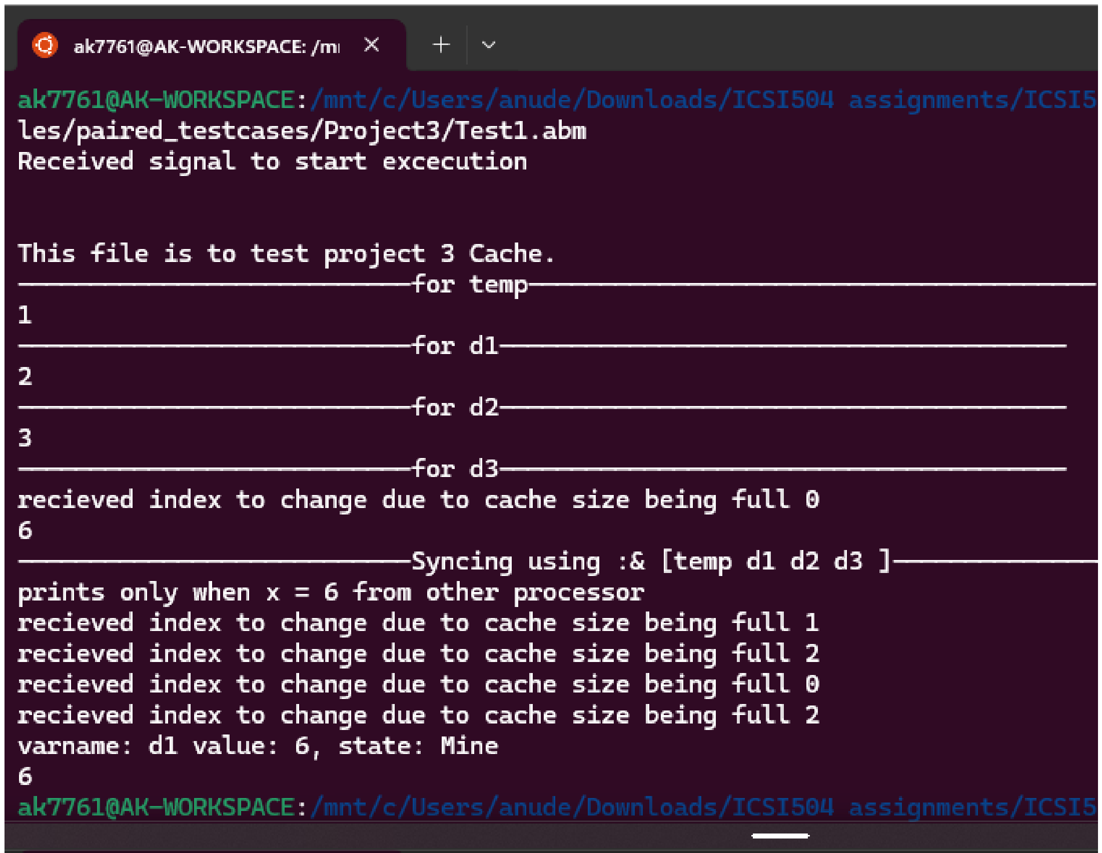


```
Address: x, Value: 1, Syncedaddress: , Status: Mine TimeStamp: 5720 seconds, 591968401 na
econds
Address: f, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5720 seconds, 640426901 
oseconds


did come here in to read

in Cache the value of b is -2147483648

after reading: b, 0

Address: x, Value: 1, Syncedaddress: , Status: Mine TimeStamp: 5720 seconds, 591968401 na
econds
Address: f, Value: 1, Syncedaddress: , Status: Mine TimeStamp: 5720 seconds, 640879301 na
econds
Address: b, Value: 0, Syncedaddress: , Status: Shared TimeStamp: 5720 seconds, 690347201 
oseconds


results in 1 and 1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project$
```

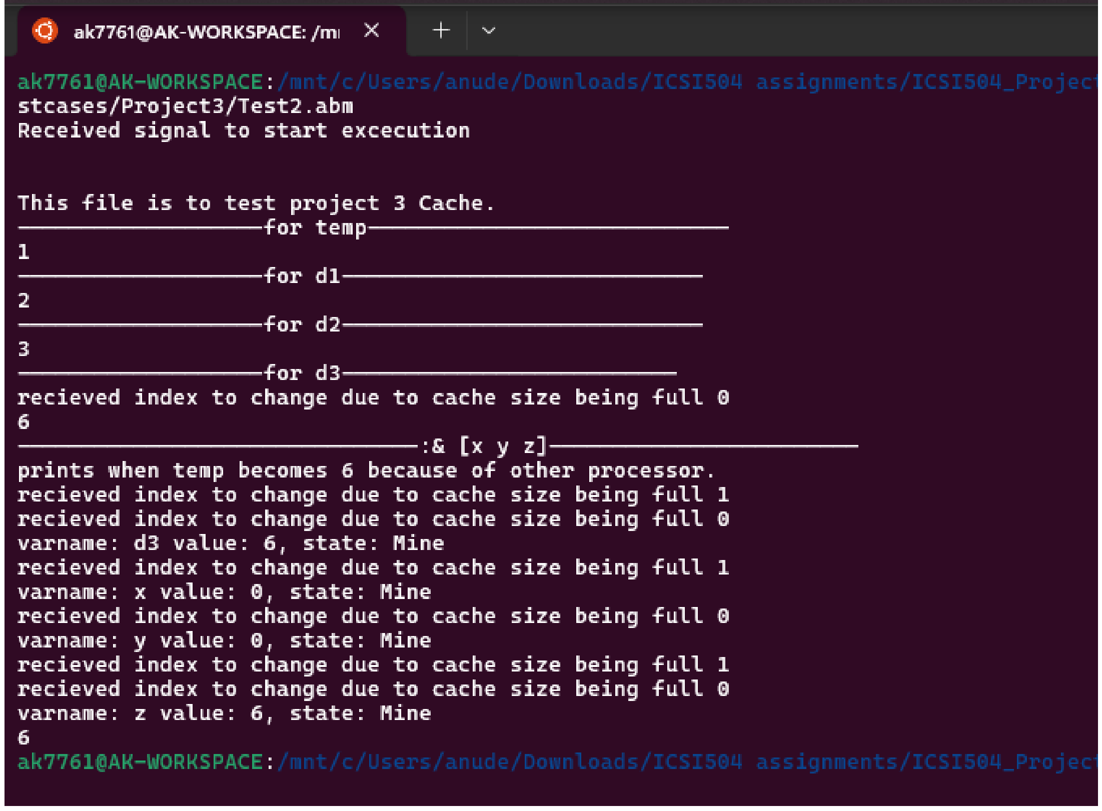


```
did come here in to read

in Cache the value of b is 0

I did come here to read bdid come here in to read

in Cache the value of x is -2147483648

after reading: x, 1

Address: f, Value: 0, Syncedaddress: , Status: Initialize_Invalid TimeStamp: 5720 seconds
91566401 nanoseconds
Address: b, Value: 1, Syncedaddress: , Status: Shared TimeStamp: 5720 seconds, 691044701 
oseconds
Address: x, Value: 1, Syncedaddress: , Status: Shared TimeStamp: 5720 seconds, 691731101 
oseconds


0
1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project$
```

i *Project3 Own testcase – result 6 is expected:*



```
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI5
les/paired_testcases/Project3/Test1.abm
Received signal to start excecution


This file is to test project 3 Cache.
----------------------------------for temp-----------------------------------------
1
----------------------------------for d1------------------------------------------
2
----------------------------------for d2------------------------------------------
3
----------------------------------for d3------------------------------------------
recieved index to change due to cache size being full 0
6
-------------------------------Syncing using :& [temp d1 d2 d3 ]---------------
prints only when x = 6 from other processor
recieved index to change due to cache size being full 1
recieved index to change due to cache size being full 2
recieved index to change due to cache size being full 0
recieved index to change due to cache size being full 2
varname: d1 value: 6, state: Mine
6
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI5
```



```
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project
stcases/Project3/Test2.abm
Received signal to start excecution


This file is to test project 3 Cache.
-------------------for temp----------------------------
1
-------------------for d1----------------------------
2
-------------------for d2----------------------------
3
-------------------for d3--------------------------
recieved index to change due to cache size being full 0
6
--------------------------------:& [x y z]---------------------------
prints when temp becomes 6 because of other processor.
recieved index to change due to cache size being full 1
recieved index to change due to cache size being full 0
varname: d3 value: 6, state: Mine
recieved index to change due to cache size being full 1
varname: x value: 0, state: Mine
recieved index to change due to cache size being full 0
varname: y value: 0, state: Mine
recieved index to change due to cache size being full 1
recieved index to change due to cache size being full 0
varname: z value: 6, state: Mine
6
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project
```

b. Bugs and errors in code:

Currently the code doesn't have any bugs or errors as mentioned above it gives the desired output perfectly for the given input of instructions. All test cases passed and are working great.

c. Executable Version of solution:

Due to too many headers and .c files, I was unable to paste it here, please check it in my submission. There are more than 1500+ lines of code, please feel free to browse through given files.

d. Difficulties and solutions involved in creating solutions:

After solving Project 2 with all the states and stuff, it wasn't a huge issue to solve this and it was completely under the arc of understanding and made things very easy, Thanks to Professor Jackson.

<u>**Algorithm and datastructures:**</u>

a <u>Data Structures</u>:

- i *<u>IndexKeywordCommandPair</u>*<u>:</u> a struct that has an int line number, char arrays.

  keyword and command which makes this structure an Index Pair. This is used to store the file as this way of reading it makes it easier to collect labels, jump to target, understand scope.
- ii *<u>Stack</u>*<u>:</u> a struct using a char array as stack to pop/push/peek elements. Is used as the

  given abm functions depend on stack-based operations. Apart from this I used it in storing the last index if a call is evoked to jump back once returned.
- iii *<u>Variablearray</u>*<u>:</u> uses a 2D array to store variable names in them. This is the efficient

  way as I can get the address of variable name, and as string in C are char* arrays I used 2D array.
- iv *<u>Map</u>*<u>:</u> a basic key to value map to store variable addresses and values. This also is the

  most feasible option for the way I was solving as this makes it easier for me to read find the address of a variable name and return its value.
- v *<u>VariableManger:</u>* It layered over 2D array (Stringarray from *Variablearray.h*), and a

  Map (Map from *addresstovaluedict*) to store variables and manages its scope. This is used to make a new scope leading to a new 2D array and map every time a new scope is made. This helps me to manage both 2D array and Map together and decide. accessibility of a variable according to the scope.

Vi. CacheMemory: a struct built on Map that consists of information regarding Cache with max size 3, it updates cache based on least reused index and when removing if a particular element state is in M it will write to bus.

The above-mentioned structures are mostly derived from arrays and customized to their desired characteristics. They all are chosen intuitively to aid solving the given problem in the most modular way I can.

<u>User Documentation:</u>

a <u>What operation system used</u>: UBUNTU - 22.04.2 LTS

b <u>How to compile your code:</u>

After having all the .h and .c files,
In ubuntu terminal use for each
<u>For the client:</u>
gcc Stack.c ABMMethods.c StackBasedMicroprocessor.c ABMReader.c Variablearray.c addresstovaluedict.c IndexKeywordCommandPair.c VariableManager.c CacheMemory.c -o Project3
<u>For server which is also memory bus:</u>

gcc -o server Variablearray.c addresstovaluedict.c VariableManager.c DuoProcessor_server.c -lpthread

which compiles my code.

c <u>What other applications are required to run:</u> NONE.

d <u>How to Run your program with parameters:</u>

After compiling my code as mentioned above,

Start the server first: ./server

if you root into **Project** and enter the abm filenames in different terminals that must be executed

./Project3 “file-name” ***=> ex: ./Project1 abmfiles/test1.abm → terminal 1***
./Project3 “file-name” ***=> ex: ./Project1 abmfiles/test2.abm → terminal 2***

You should now see the desired output printed in the terminal.

e Other requirements: NONE.