PROJECT 2

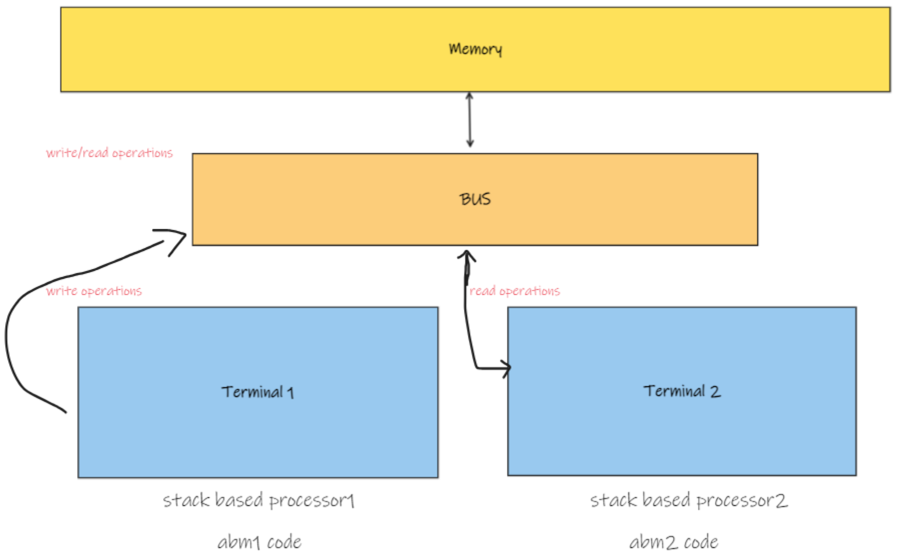
Sai vishnu Anudeep Kadiyala
001450445

NOVEMBER 7TH, 2023

TABLE OF CONTENTS:

System Documentation:

a High-level dataflow diagram.



As mentioned above in the diagram, we have two terminals running concurrently using stackbased microprocessor, and we have a memory bus that waits till connection establishes, and once established uses write and read operations on shared memory for shared variables.

b List of Routines and descriptions:

1. ABMMethods.c

Routine Name	**Routine Description**
Send to bus	Takes in the message and formats accordingly to send it to the bus

Receive from bus	Once called keeps listening to the bus and once a signal received will return that value.

2 VariableManager.c

Routine Name	**Routine Description**
FindInGlobalScope	Gets the value from global scope
FindInGlobalContainerbyaddress	Returns value using global address.
getaddressfromGlobalContainer	Gets address for a given variable name from global container.
getnameof_variable_byaddress_fromGlobalContainer	For a given address returns its variable name.
getVariableaddressByOffset	For a given variable name + offset it will return the variable name which offset far from given variable name if there exist one.
setSyncBetween	Sets sync between address => :&
InGlobalScopeSetStatus	sets the state of the variable
InGlobalScopeFindStatus	returns the state of the variable
InGlobalScopeIsItGivenStatus	Checks if the given state is the state of the variable
InGlobalScopeFindSyncedWith	Returns synced address
InGlobalsFindSyncedInstances	Returns address that are synced with this.

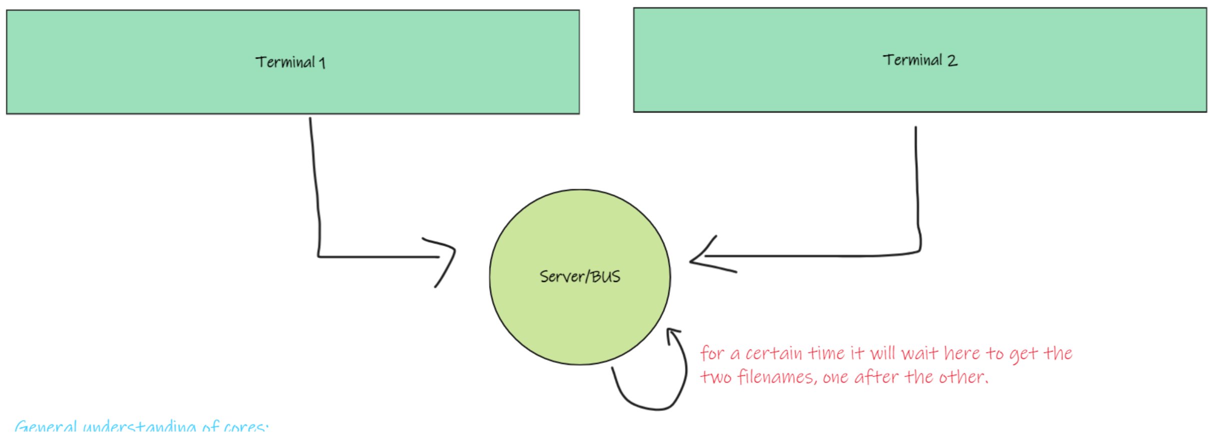
3 DuoCoreprocessor.c

Routine Name	**Routine Description**
Connectionhandler	Handlers both the connections by threading for them
Signal handler	Handles signal for the both connections until halt in both the connections abm file is encountered.

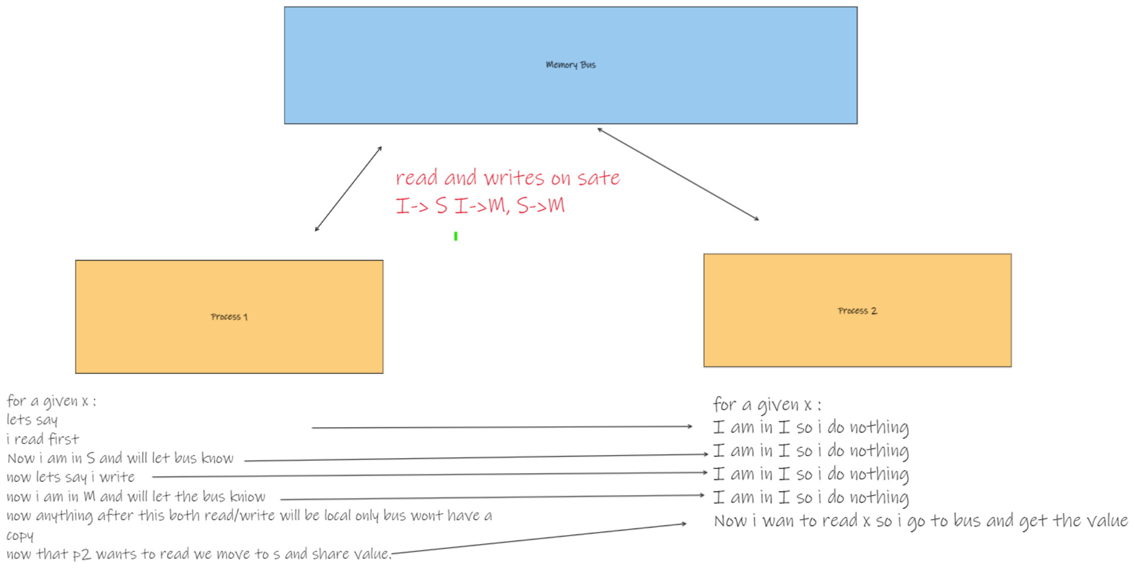
c <u>Implementation Method:</u>

I have used socket programming to connect two terminals to the bus such that every time when I make a request it comes through socket to read or write for each core processor.

The bus now is a state machine which, based on reads and writes, sends signals to the cores to change state of the machines accordingly. The bus also makes sure to lock the memory address so that there won't be any clashes and synchronization is maintained continuously.


Terminal 1
Terminal 2
Server/BUS
for a certain time it will wait here to get the two filenames, one after the other.
General understanding of cores:

So here we use sockets to create multiple sockets, at most 2, and then using the multi-threading concept we signal them to start execution which happens almost at the same time forcing both the threads to run at same time. Now considering an example if we have a X as global variable that is shared between P1 and P2, on concurrent running. It will be initially in state I and once a read happens on through bus it goes to S and stays there, while it is in S it will stay there for the rest of reads, until another processor writes. If a write happens, we move to state M where we own the variable and rest all processors move their respective states to I. while one is in M and any other processor wants to read, the bus informs the processors to share the value which will be state S and there by both the processors will hold same value.


Memory Bus
read and writes on sate
I-> S I->M, S->M
Process 1
Process 2
for a given x :
lets say
i read first
Now i am in S and will let bus know
now lets say i write
now i am in M and will let the bus kniow
now anything after this both read/write will be local only bus wont have a copy
now that p2 wants to read we move to s and share value.
for a given x :
I am in I so i do nothing
I am in I so i do nothing
I am in I so i do nothing
I am in I so i do nothing
Now i wan to read x so i go to bus and get the value

The above is a simplest example of a variable x and what it goes through to establish concurrency. While this explains the story of the bus we have & which will be maintained in the variable container, such that every variable has synced addresses, that makes every update on a variable reflecting on its synced address.

Similarly, the override +/- are managed using the memory such that if a given address + offset have a value we then use that address else if it is NULL, we exit the program.

Test Documentation:

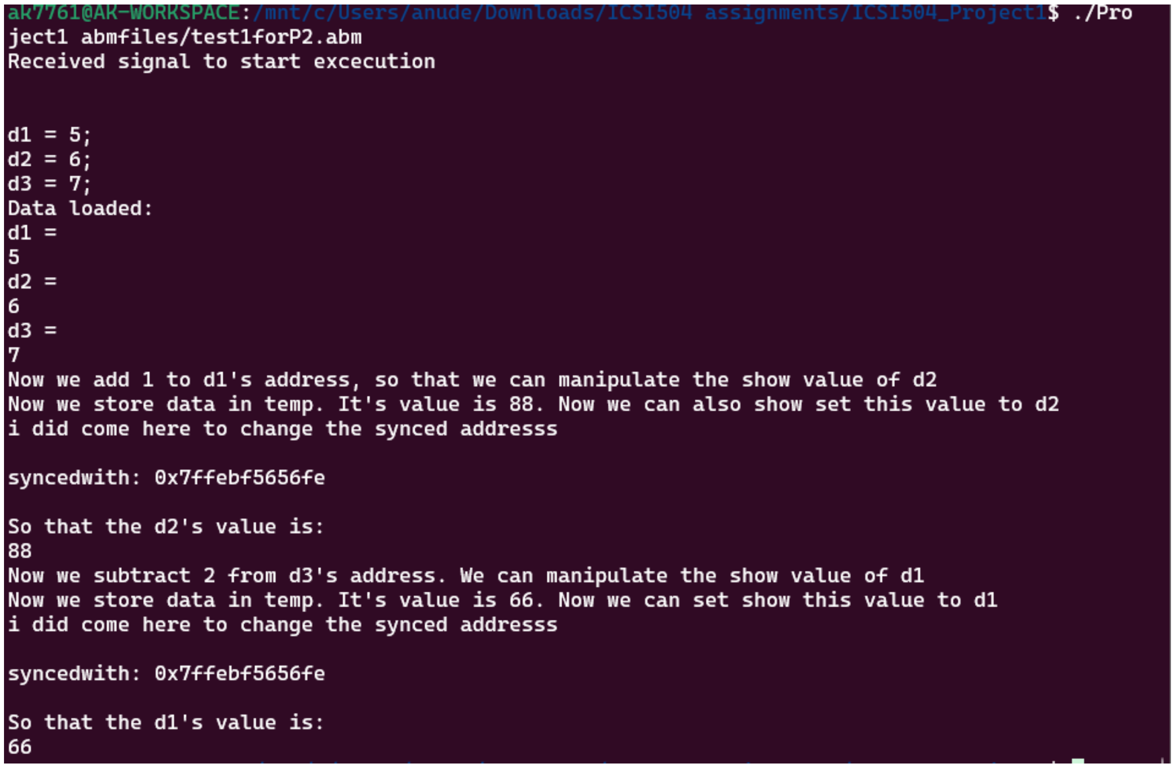
a How Did I Test My Program:

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

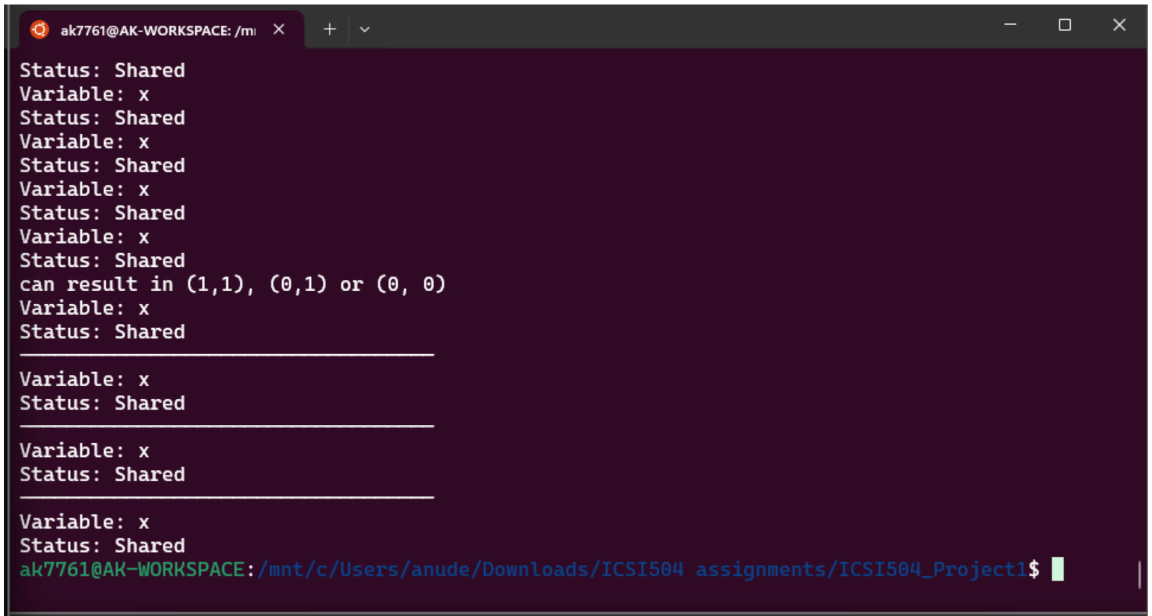
Below are the screenshots of outputs for each file. To mention they have matched exactly with each of the .out files provided showing the provided solutions credibility.

i *Testforp2.abm:*

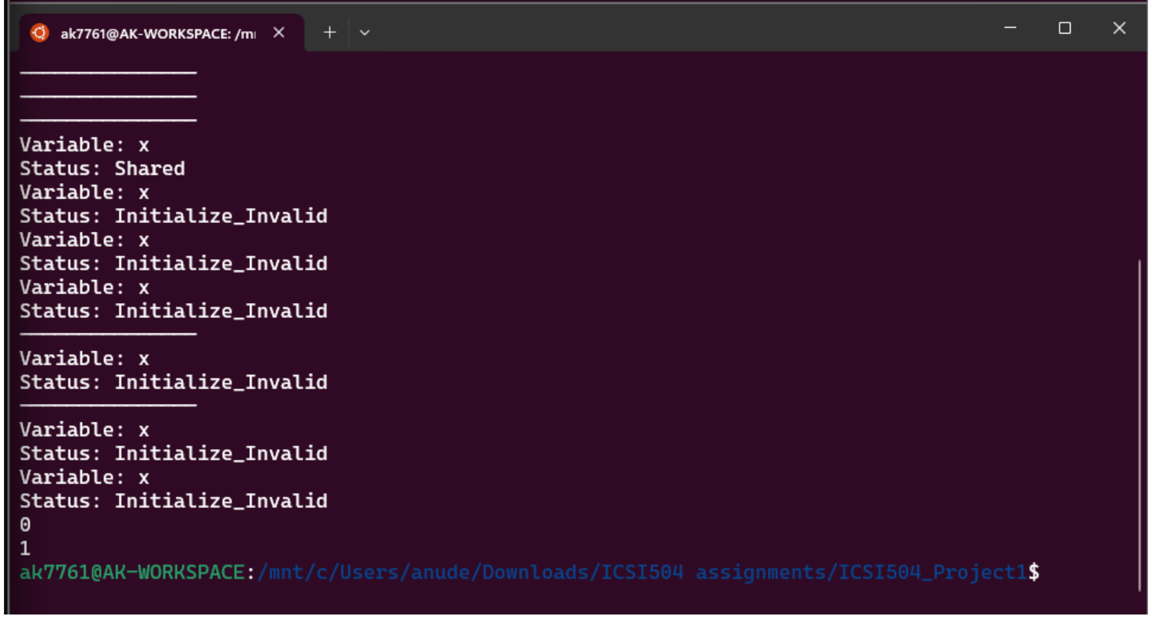


```
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/test1forP2.abm
Received signal to start excecution


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:*

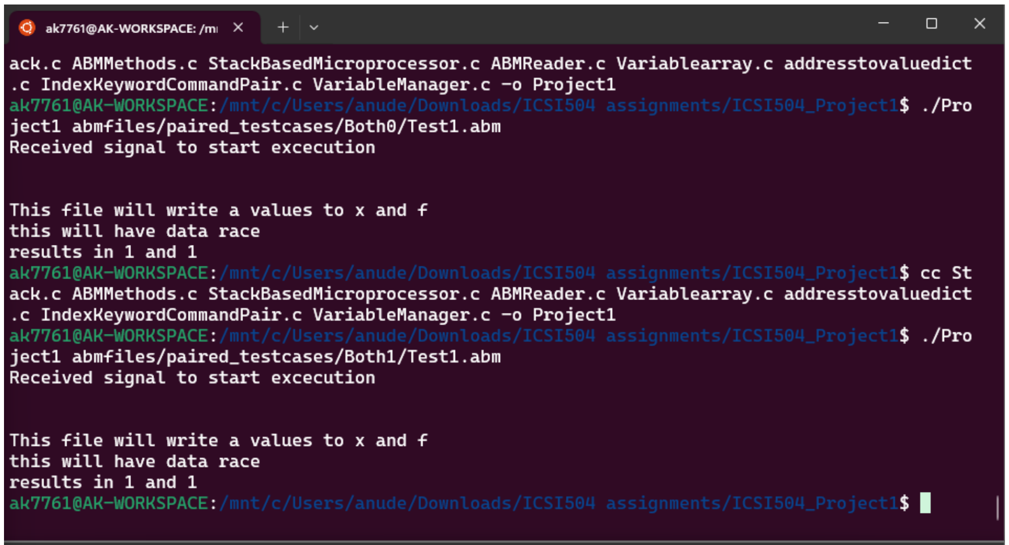


```
Status: Shared
Variable: x
Status: Shared
Variable: x
Status: Shared
Variable: x
Status: Shared
Variable: x
Status: Shared
can result in (1,1), (0,1) or (0, 0)
Variable: x
Status: Shared
___________________________________
Variable: x
Status: Shared
___________________________________
Variable: x
Status: Shared
___________________________________
Variable: x
Status: Shared
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$
```

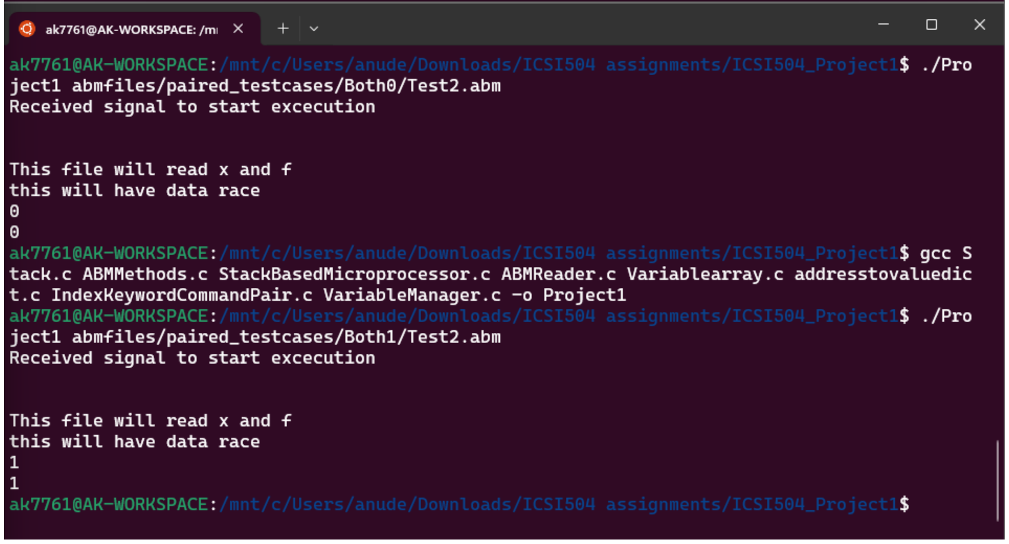


```
_______________
_______________
_______________
Variable: x
Status: Shared
Variable: x
Status: Initialize_Invalid
Variable: x
Status: Initialize_Invalid
Variable: x
Status: Initialize_Invalid
_______________
Variable: x
Status: Initialize_Invalid
_______________
Variable: x
Status: Initialize_Invalid
Variable: x
Status: Initialize_Invalid
0
1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$
```

i *Bothof them are 1 case:*

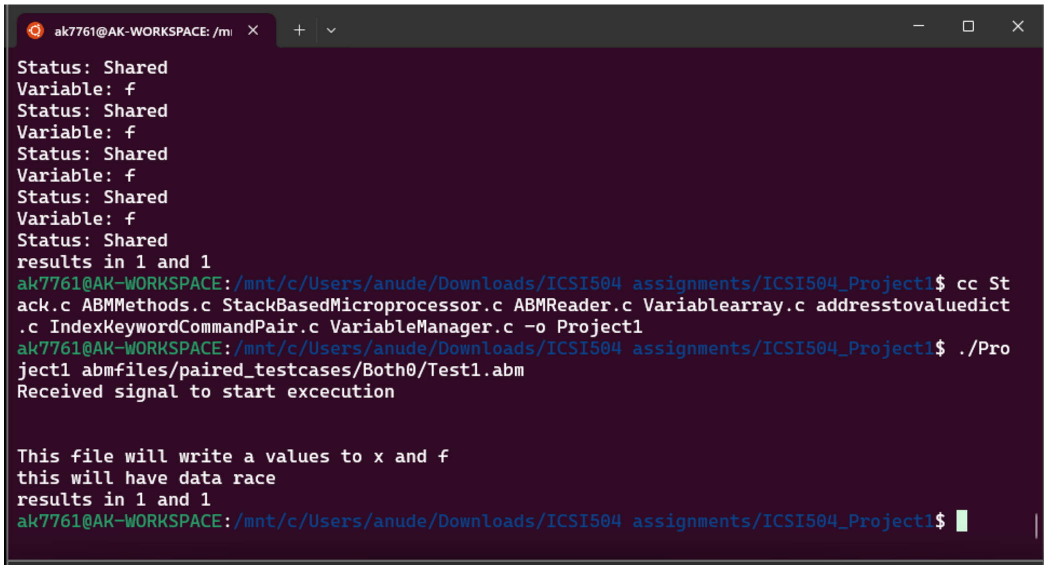


```
ack.c ABMMethods.c StackBasedMicroprocessor.c ABMReader.c Variablearray.c addresstovaluedict
.c IndexKeywordCommandPair.c VariableManager.c -o Project1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/Both0/Test1.abm
Received signal to start excecution


This file will write a values to x and f
this will have data race
results in 1 and 1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ cc St
ack.c ABMMethods.c StackBasedMicroprocessor.c ABMReader.c Variablearray.c addresstovaluedict
.c IndexKeywordCommandPair.c VariableManager.c -o Project1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/Both1/Test1.abm
Received signal to start excecution


This file will write a values to x and f
this will have data race
results in 1 and 1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$
```

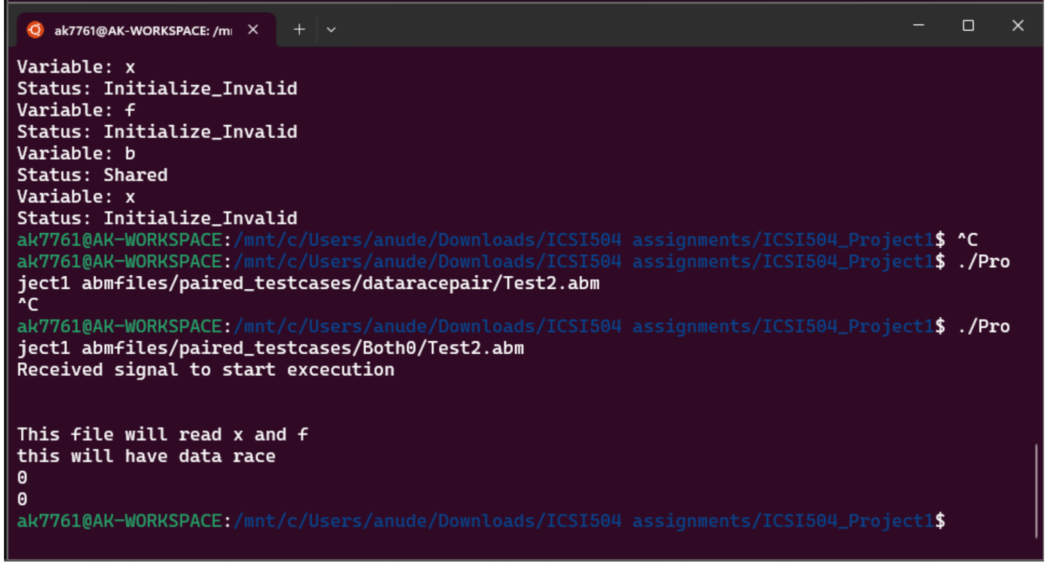


```
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/Both0/Test2.abm
Received signal to start excecution


This file will read x and f
this will have data race
0
0
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ gcc S
tack.c ABMMethods.c StackBasedMicroprocessor.c ABMReader.c Variablearray.c addresstovaluedic
t.c IndexKeywordCommandPair.c VariableManager.c -o Project1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/Both1/Test2.abm
Received signal to start excecution


This file will read x and f
this will have data race
1
1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$
```

iv. *Both are 0 case :*

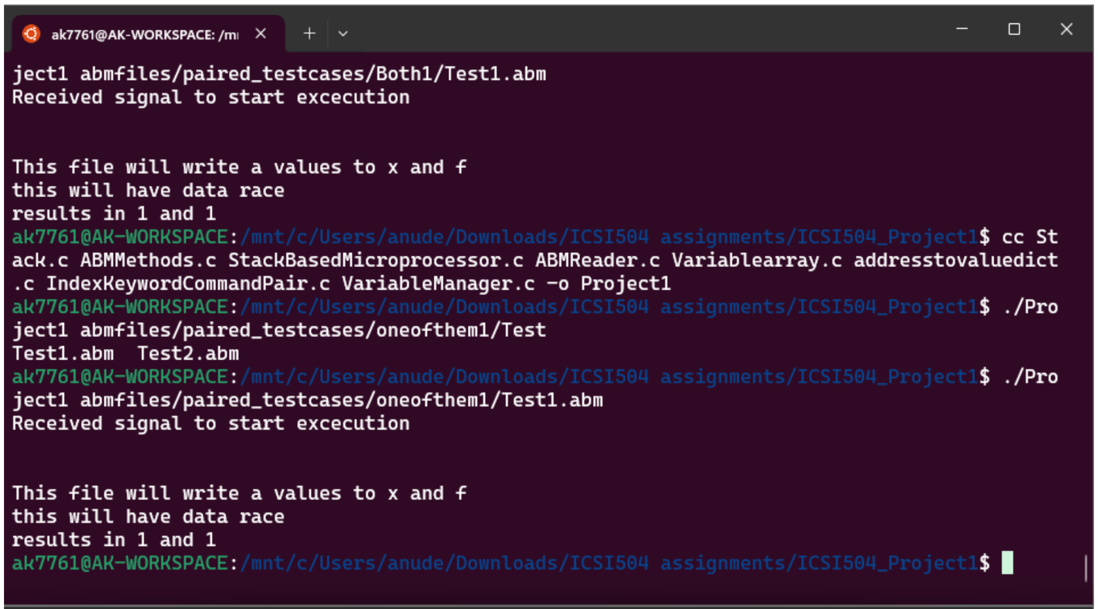


```
Status: Shared
Variable: f
Status: Shared
Variable: f
Status: Shared
Variable: f
Status: Shared
Variable: f
Status: Shared
results in 1 and 1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ cc St
ack.c ABMMethods.c StackBasedMicroprocessor.c ABMReader.c Variablearray.c addresstovaluedict
.c IndexKeywordCommandPair.c VariableManager.c -o Project1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/Both0/Test1.abm
Received signal to start excecution


This file will write a values to x and f
this will have data race
results in 1 and 1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$
```

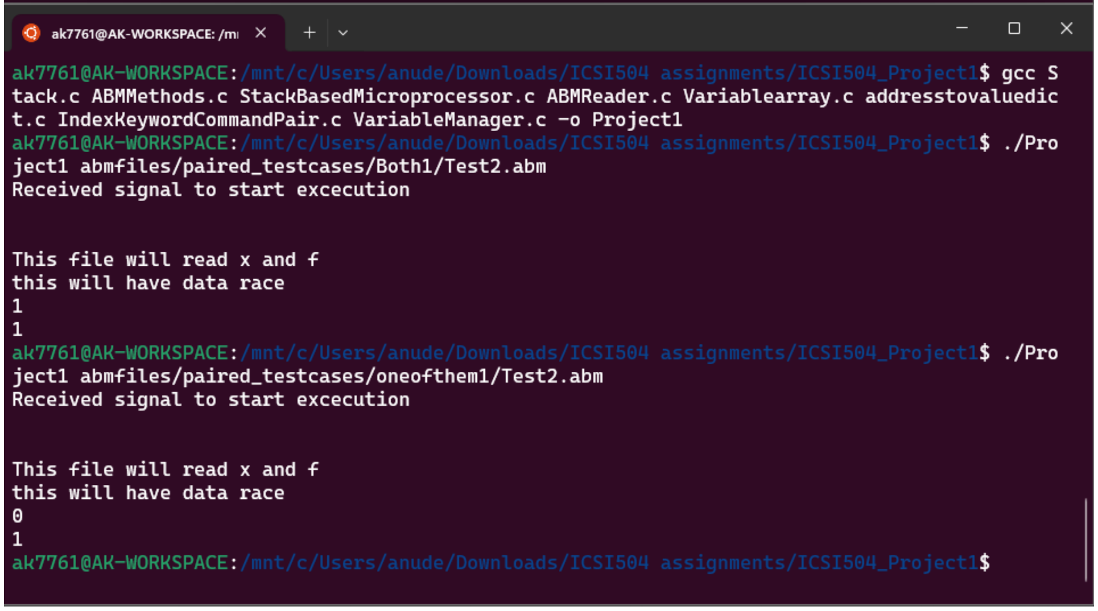


```
Variable: x
Status: Initialize_Invalid
Variable: f
Status: Initialize_Invalid
Variable: b
Status: Shared
Variable: x
Status: Initialize_Invalid
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ^C
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/dataracepair/Test2.abm
^C
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/Both0/Test2.abm
Received signal to start excecution


This file will read x and f
this will have data race
0
0
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$
```

v *One of them Is 1:*



```
ject1 abmfiles/paired_testcases/Both1/Test1.abm
Received signal to start excecution


This file will write a values to x and f
this will have data race
results in 1 and 1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ cc St
ack.c ABMMethods.c StackBasedMicroprocessor.c ABMReader.c Variablearray.c addresstovaluedict
.c IndexKeywordCommandPair.c VariableManager.c -o Project1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/oneofthem1/Test
Test1.abm  Test2.abm
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/oneofthem1/Test1.abm
Received signal to start excecution


This file will write a values to x and f
this will have data race
results in 1 and 1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$
```



```
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ gcc S
tack.c ABMMethods.c StackBasedMicroprocessor.c ABMReader.c Variablearray.c addresstovaluedic
t.c IndexKeywordCommandPair.c VariableManager.c -o Project1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/Both1/Test2.abm
Received signal to start excecution


This file will read x and f
this will have data race
1
1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$ ./Pro
ject1 abmfiles/paired_testcases/oneofthem1/Test2.abm
Received signal to start excecution


This file will read x and f
this will have data race
0
1
ak7761@AK-WORKSPACE:/mnt/c/Users/anude/Downloads/ICSI504 assignments/ICSI504_Project1$
```

b <u>Bugs and errors in code:</u>

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 working great

c <u>Executable Version of solution:</u>

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

d <u>Difficulties and solutions involved in creating solutions:</u>

The overall assignment is very challenging starting from concurrent run of programs to the last bit of address assignment. It was very heavily intuitive and caused a lot of questions and discussions. I would thank professor for his help of this.

Algorithm and datastructures:

a Data Structures:

- i *IndexKeywordCommandPair*: 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 *Stack*: 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 *Variablearray*: 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. *Map*: 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. *VariableManger:* 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.

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.

User Documentation:

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 -o Project2

<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 **Project2** and enter the abm filenames in different terminals that must be executed

./Project2 "file-name" ***=> ex: ./Project1 abmfiles/test1.abm → terminal 1***
./Project2 "file-name" ***=> ex: ./Project1 abmfiles/test2.abm → terminal 2***

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

e Other requirements: NONE.