Test Case 1:

```
.text:
@8
ADDI R1, R2, 50
ADDI R1, R1, 3
LW R4, R1, 0
MUL R2, R4, R3
SW R2, R1, 0
BEQ R2, R5, 12
ADDI R2, R2, R1
.data:
@50 12
@53 13
@70 14
```

Test Case 2:

```
For(int i=0;i<5;i++)
{
        A[i+1]=A[i]+!;
}
```

```
.text:
@8
ADDI R1, R1, 20
ADDI R2, R2, 0
Loop:
ADD R3, R4, R2
LW R3, R3, 0
ADDI R4, R3, 1
ADDI R6, R2, 4
ADD R3, R4, R6
SW R4, 0(R3)
ADDI R2, R2, 4
BEQ R2, R1, EXIT
JMP Loop
EXIT:
.data:
@0 12
@4 17
@8 18
@10 36
@17 82
```

Test Case 3:

```
For(int i=0;i<4;i++)
{
A[i]=A[i+!]+B[i];
}
```

```
.text:
@8
```

```
ADDI R1, R0, 20
ADDI R2, R0, 0
ADDI R4, R0, 0
Loop:
ADD R3, R4, R2
ADD R8, R5, R2
LW R3, R3, 0
ADDI R3, R3, 1
ADDI R6, R2, 4
LW R7, R8, 0
ADD R3, R7, R3
ADD R7, R4, R6
SW R3, R8, 0
ADDI R2, R2, 4
BEQ R2, R1, EXIT
JMP Loop
EXIT:
.data:
@0 12
@4 17
@36 18
@60 36
@72 82
```

Test Case 4:

```
.text:
@0
ADDI R2, R2, 10
ADD R2, R2, R3
SUB R3, R3, R2
NAND R5, R3, R1
JMP TRYIT
JMP EXIT
TRYIT:
MUL R3, R5, R3
ADDI R7, R0, 5
RET R7
EXIT:
ADD R0, R0, R0
```

Test Case 5:

```
.text:
@0
ADDI R6, R0, 4
ADDI R7, R0, 1
ADD R1, R0, R0
ADDI R2, R0, 50
ADDI R4, R0, 30
LOOP:
BEQ R6, R0, END
ADD R1, R1, R6
SUB R6, R6, R7
```

```
SW R1, R2, 50
ADD R5, R2, R4
LW R3, R5, 0
ADDI R2, R2, 2
JMP LOOP
END:
ADD R0, R0, R0
.data:
@80 30
@82 32
@84 34
@86 36
```

Test Case 6:

.text:
@0
ADDI R1, R0, 10
ADD R2, R1, R3
SW R2, R1, 0
LW R2 , R1 , 0
SW R2, R1, 0
LW R3 , R1 , 0
BEQ R2, R3, JUMP
SUB R2, R1, R3
LW R2 , R1 , 0
SW R2, R1, 0
ADD R2, R2, R2
JUMP:
ADD R3, R2, R3
LW R2 , R1 , 0
SW R2, R1, 0

Test Case 7:

X[5]= X[2*j-i];

Base address = r3
J = r1
I = r2

This code will load from memory[53] and store into memory[55]

.text:
@0
ADDI R3, R0, 50
ADDI R1, R0, 2
ADDI R2, R0, 1
ADD R4, R1, R1
SUB R4, R4, R2
ADD R4, R4, R3
LW R5, R4, 0

```
SW R5, R3, 5
.data:
@53 10
```

Test Case 8:

while (save[j] != -k) j++;

J = R1
K = R2 = 10
BASE ADDRESS = R6

```
.text:
@0
ADDI R2, R0, 10
SUB R3, R0, R2
ADDI R6, R0, 50
ADDI R1, R1, 0
Loop:
ADD R5, R1, R6
LW R5, R5, 0
BEQ R5, R3, Exit
ADDI R1, R1, 1
JMP Loop
Exit:
.data:
@50 2
@51 3
@52 -10
@53 7
```

Test Case 9:

```
.text:
@0
ADDI R4, R0, 200
ADDI R3, R0, 5
ADDI R6, R0, 14
JALR R1, R6
LAlel:
SW R3, R4, 0
BEQ R0, R0, hi
LW R5, R4, 0
label:
SW R3, R4, 0
ADDI R6, R0, 6
ADDI R7, R0, 7
hi:
ADDI R1, R0, 1
JMP babel
ADDI R2, R0, 2
babel:
```

```
JMP Exit
SUB R7, R1, R7
RET R1
Exit:
ADDI R2, R0, 3
.data:
@200 10
```

Test Case 10:

```
Int x = 50, i=0;
Do {
        Result += M[i];
        i++;
        result+=M[i];
        i++;
        x--;
}while (x!=0)
```

BASE ADDRESS -> R4
X -> R1
RESULT -> R3

```
.text:
@0
ADDI R1, R0, 4
ADDI R5, R5, 1
ADDI R4, R4, 50
LOOP:
LW R2, R4, 0
ADD R3, R3, R2
LW R2, R4, 1
ADDI R4, R4, 2
SUB R1, R1, R5
BEQ R1, R0, EXIT
JMP LOOP
EXIT:
.data:
@50 1
@51 2
@52 3
@53 4
@54 5
@55 6
@56 7
@57 8
@58 9
@59 10
@60 11
```

Test Case 11: (Dr's)

```
.text:
```

```
@0
LW R1, R0, 300
LW R2, R0, 301
LW R3, R0, 302
MUL R4, R2, R3
MUL R5, R1, R2
LW R6, R0, 303
ADD R6, R3, R6
ADDI R7, R7, 8
SUB R3, R6, R7
SW R6, R0, 303
.data:
@300 19
@301 2
@302 25
@303 208
```

Test Case 12: (Dr's)

```
.text:
@0
LW R1, R0, 100
ADDI R2, R0, 101
ADD R3, R2, R1
ADDI R4, R0, 0
L1:
BEQ R4, R1, L2
LW R5, R2, 0
MUL R5, R5, R1
SW R5, R3, 0
ADDI R2, R2, 1
ADDI R3, R3, 1
ADDI R4, R4, 1
JMP L1
L2:
ADD R0, R0, R0
.data:
@100 3
@101 12
@102 -5
@103 7
```

Test Case 13:

```
.text:
@0
ADDI R1, R0, 10
ADDI R2, R0, 20
BEQ R1, R2, L1:
ADDI R3, R0, 2
MUL R1, R1, R3
ADDI R4, R1, 0
```

```
L1:
ADD R4, R4, R3
ADDI R5, R0, 26
SUB R6, R4, R5
BEQ R6, R0, L2
JMP L1
ADDI R6, R0, 6
L2:
NAND R7, R6, R3
```

Testcase 14:

```
.text:
@0
ADDI R1, R0, 1
ADDI R2, R0, 2
L1:
ADDI R4, R0, 4
ADDI R5, R0, 5
ADDI R6, R0, 6
ADDI R7, R0, 7
BEQ R2, R1, L1
ADDI R3, R0, 1
BEQ R3, R1, L2
ADDI R4, R0, 2
L2:
ADDI R5, R0, 9
```

Testcase 15:

```
.text:
@0
ADDI R6, R0, 100
ADDI R1, R0, 0
ADDI R2, R0, 2
ADDI R3, R0, 3
L1:
BEQ R1, R6, EXIT
BEQ R2, R3, L2
ADDI R2, R0, 3
JMP L3
L2:
ADDI R2, R0, 2
L3:
ADDI R1, R1, 1
JMP L1
EXIT:
ADDI R7, R0, 7
```