اعضای گروه:
سوگل گودرزی ۸۱۰۱۹۸۴۶۷
عرفان پناهی ۸۱۰۱۹۸۳۶۹

*** بررسی شرایط انجام دستور (C):

در شرایط انجام دستورات نیاز است که C را بررسی کنیم. حال حالت های مختلف برای این شرایط را طبق جدول زیر تحلیل می کنیم. سپس با استفاده از K-map ، بیت D را می سازیم تا شرط انجام دستور از مسیر داده به کنترلر ارسال شود. اگر که این بیت ۱ باشد دستور انجام می شود و اگر این بیت صفر باشد دستور انجام نمی شود.

Z	N	V	C = 00	C = 01	C = 10	C = 11
0	0	0		✓		✓
0	0	1			✓	✓
0	1	0			✓	✓
0	1	1		✓		✓
1	0	0	✓			✓
1	0	1	✓		✓	✓
1	1	0	✓			✓
1	1	1	✓		✓	✓

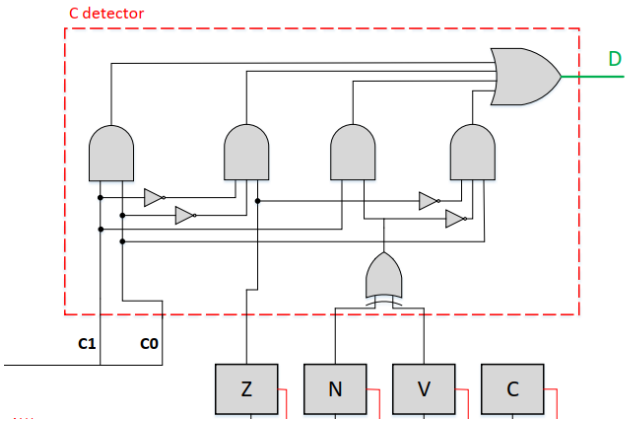
NV \ C1C0	00	01	11	10
00	0	0	0	0
01	1	0	1	0
11	1	1	1	1
10	0	1	0	1

Z=0

NV \ C1C0	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	1	1	1	1
10	0	1	0	1

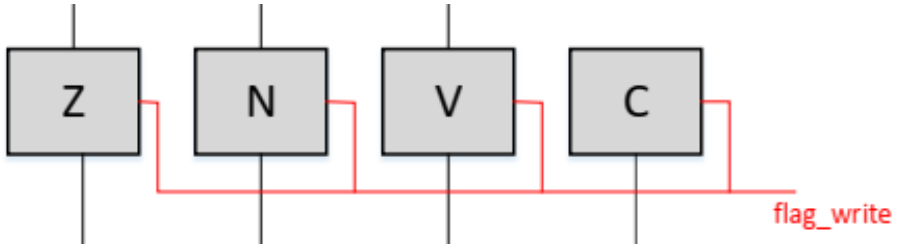
Z=1

$$D = C_1C_0 + Z\overline{C_1}.\overline{C_0} + C_1(N \oplus V) + \bar{Z}C_0(\overline{N \oplus V})$$



***** بررسی تغییرات پرچم ها:**

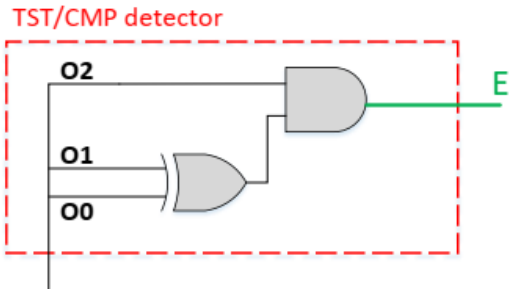
برای هر پرچم مطابق تصویر زیر یک رجیستر تک بیتی در نظر گرفته شده است که همه آنها با یک سیگنال کنترلی (flag_write) هنگام اتمام دستور پردازش داده، لود و به روز می‌شوند.



***** بررسی دستورات پردازش داده و ذخیره نهایی در رجیستر ها:**

با توجه به اینکه در دستورات TST و CMP نیازی به ذخیره کردن نتیجه در رجیستر نداریم و فقط پرچم ها برای ما مهم است، باید یک بیت E را از مسیر داده به کنترلر بدهیم تا متوجه این دو دستور شود. برای ساخت بیت E با استفاده از opc و جدول زیر، مدار زیر را طراحی می کنیم:

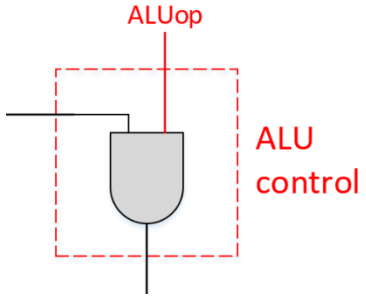
E	Instructions
0	ADD , SUB , RSB , AND , NOT , MOV
1	TST , CMP

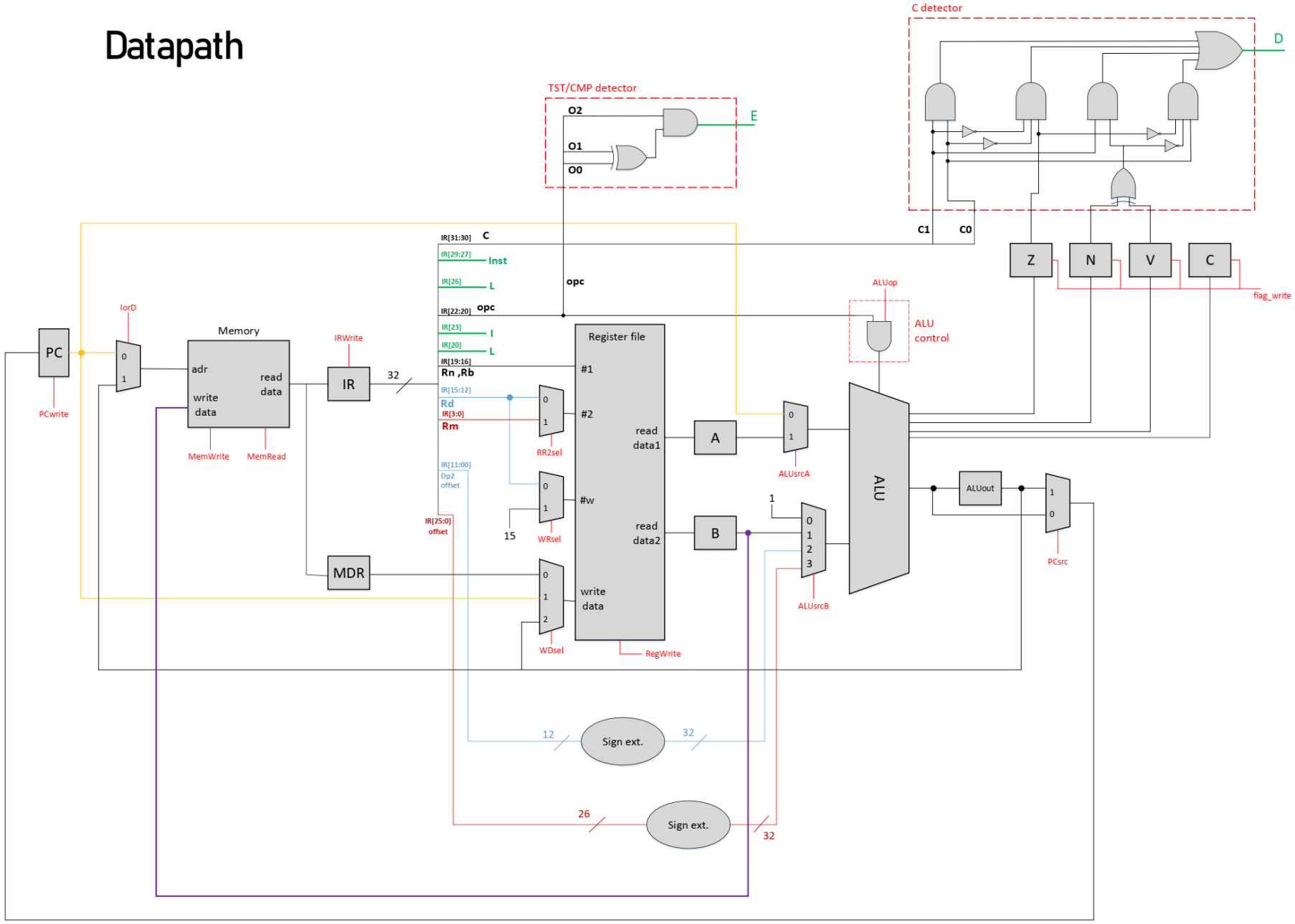


*****صرفه جویی در انتقال داده از کنترلر به مسیرداده و برعکس:**

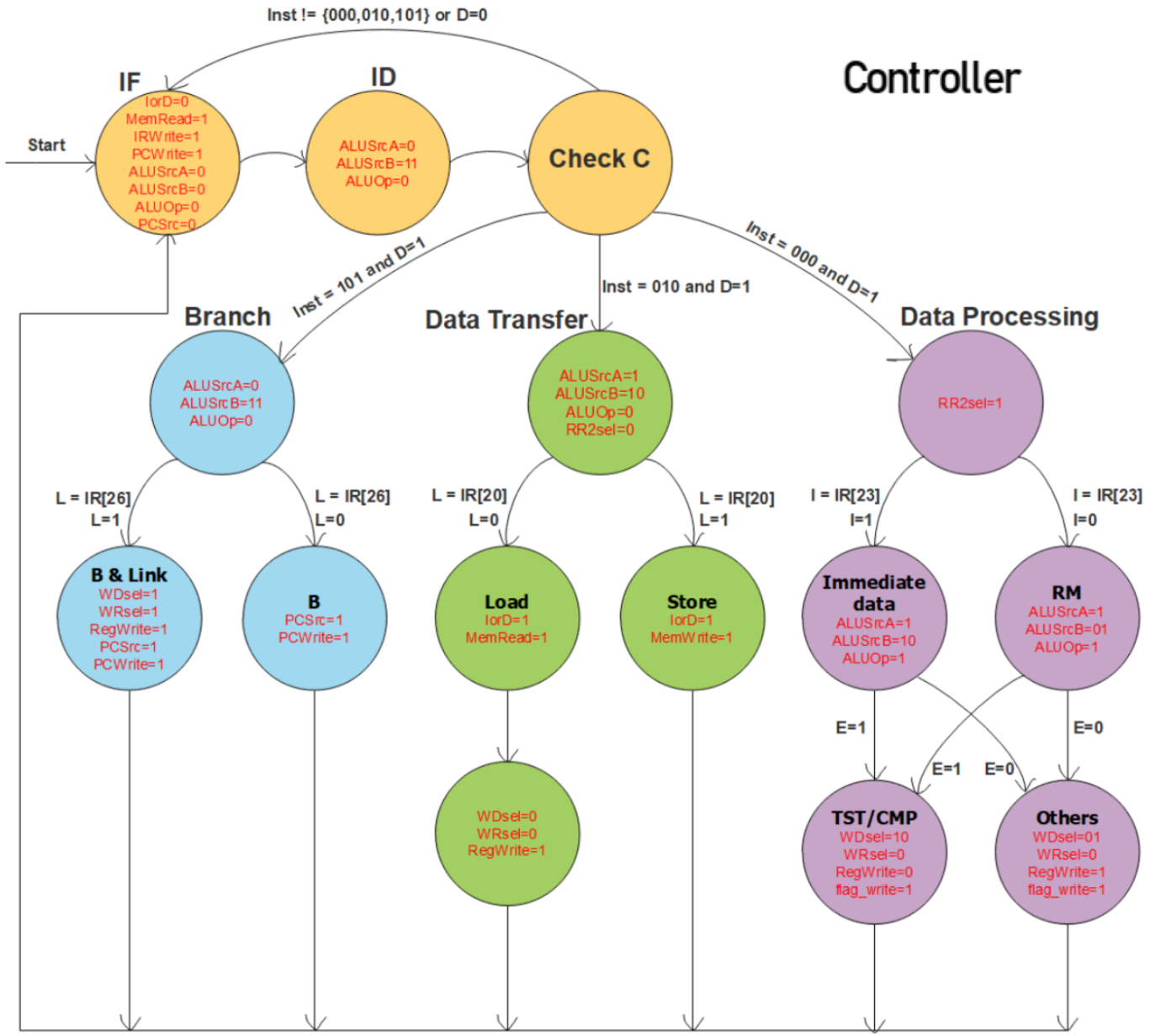
برای صرفه جویی در انتقال تعداد بیت ها از کنترلر به مسیر داده به جای انتقال سه بیت کنترل ALU ، یک بیت ALUOp تعریف کرده و با استفاده از گیت AND کنترل ALU را به صورت زیر طراحی می کنیم:

Inst = IR[29:27]	Instruction Type	ALUOp
Inst = 000	Data Processing	1
Inst = 010	Data Transfer	0
Inst = 101	Branch	0
PC = PC + 1	-	0




Datapath
C detector
D
TST/CMP detector
O2
O1
O0
E
C1
C0
Z
N
V
C
flag_write
ALUop
ALU control
IorD
PC
PCwrite
Memory
adr
write data
read data
MemWrite
MemRead
IRWrite
IR
32
IR[31:30] C
IR[29:27] Inst
IR[26] L
IR[22:20] opc
opc
IR[23] I
IR[20] L
IR[19:16] Rn ,Rb
IR[15:12] Rd
IR[3:0] Rm
IR[11:00] Op2 offset
IR[25:0] offset
Register file
#1
#2
#w
read data1
read data2
write data
RR2sel
WRsel
WDsel
RegWrite
A
B
ALUsrcA
ALUsrcB
ALU
ALUout
PCsrc
MDR
15
1
12
32
Sign ext.
26
Sign ext.
32


Datapath
Controller
Inst = IR[29:27]
L = IR[26]
I = IR[23]
L = IR[20]
E
D
flag_write
ALUop
PCWrite
PCsrc
ALUsrcA
ALUsrcB
RR2sel
WRsel
WDsel
IRWrite
MemWrite
MemRead
IorD
RegWrite


Controller
Inst != {000,010,101} or D=0
IF
IorD=0
MemRead=1
IRWrite=1
PCWrite=1
ALUSrcA=0
ALUSrcB=0
ALUOp=0
PCSrc=0
Start
ID
ALUSrcA=0
ALUSrcB=11
ALUOp=0
Check C
Inst = 101 and D=1
Inst = 010 and D=1
Inst = 000 and D=1
Branch
ALUSrcA=0
ALUSrcB=11
ALUOp=0
L = IR[26]
L=1
L = IR[26]
L=0
B & Link
WDsel=1
WRsel=1
RegWrite=1
PCSrc=1
PCWrite=1
B
PCSrc=1
PCWrite=1
Data Transfer
ALUSrcA=1
ALUSrcB=10
ALUOp=0
RR2sel=0
L = IR[20]
L=0
L = IR[20]
L=1
Load
IorD=1
MemRead=1
Store
IorD=1
MemWrite=1
WDsel=0
WRsel=0
RegWrite=1
Data Processing
RR2sel=1
I = IR[23]
I=1
I = IR[23]
I=0
Immediate data
ALUSrcA=1
ALUSrcB=10
ALUOp=1
RM
ALUSrcA=1
ALUSrcB=01
ALUOp=1
E=1
E=0
E=1
E=0
TST/CMP
WDsel=10
WRsel=0
RegWrite=0
flag_write=1
Others
WDsel=01
WRsel=0
RegWrite=1
flag_write=1

* الگوریتم پیدا کردن کوچکترین مقدار یک آرایه ۱۰ عنصری و اندیس آن:

```
array A[0: 9];
Value = A[0];
Index =0;
for (int i=1;i<10;i++){
        if(Value > A[i]){
                Value = A[i];
                Index = i;
        }
}
//Value: Minimum value of A
//Index: Index of Minimum value of A
```

❊ برنامه با زبان اسمبلی و زبان ماشین:

برای انجام دستورات مناسب به منظور اینکه کوچکترین مقدار یک آرایه ۱۰ عنصری و همچنین اندیس آن را بیابیم و در آدرس های ۲۰۰۰ و ۲۰۰۴ حافظه بنویسیم، دستورات زیر را ابتدا به زبان کد ماشین نوشته سپس معادل باینری آن ها را پیدا کرده ایم و در آخر نیز آن ها را به اعداد هگزادسیمال تبدیل کرده و در فایلMemory.txt ذخیره سازی می کنیم:

❊ اسمبلی:

```
//R2=0 index
//R10 index of loop
Loop:          R1 = Mem[R2 +1000] //c=11
               R11 = R10 - 10 //c=11
               Jump to[PC+5] //c=00
               R3 = Mem[R10 + 1000] //c=11
               R10 = R10 + 1 //c=11
               R11 = R3 - R1 //c=11
               R2 = R10 - 1 //c=10
               Jump to [PC - 8] //c=11
EndLoop:       Mem[R0+2000] = R1 //c=11
               Mem[R0+2004] = R2 //c=11
```

❊ ماشین باینری و هگزادسیمال:

Adrress	Instructions	Binary Instructions	Hexadecimal Instructions
Adr1:	11-010000000-0-0010-0001-001111101000	1101-0000-0000-0010-0001-0011-1110-1000	d00213e8
Adr2:	11-000000-1-001-1010-1011-000000001010	1100-0000-1001-1010-1011-0000-0000-1010	c09ab00a
Adr3:	00-101-0-00000000000000000000000101	0010-1000-0000-0000-0000-0000-0000-0101	28000005
Adr4:	11-010000000-0-1010-0011-0001111101000	1101-0000-0000-1010-0011-0011-1110-1000	d00a33e8
Adr5:	11-000000-1-000-1010-1010-000000000001	1100-0000-1000-1010-1010-0000-0000-0001	c08aa001
Adr6:	11-000000-0-001-0011-1011-00000000-0001	1100-0000-0001-0011-1011-0000-0000-0001	c013b001
Adr7:	10-000000-1-000-1010-0010-111111111111	1000-0000-1000-1010-0010-1111-1111-1111	808a2fff
Adr8:	11-101-0-11111111111111111111111000	1110-1011-1111-1111-1111-1111-1111-1000	ebfffff8
Adr9:	11-010000000-1-0000-0001-011111010000	1101-0000-0001-0000-0001-0111-1101-0000	d01017d0
Adr10:	11-010000000-1-0000-0010-011111010100	1101-0000-0001-0000-0010-0111-1101-0100	d01027d4

❊ داده های تست:

درنهایت نیز داده هایی بعنوان مثال مانند تصویر زیر به برنامه می دهیم و خروجی را مشاهده می کنیم که نشان می دهد برنامه به درستی کار میکند:

Test Case #1

Memory.txt

996	xxxxxxxx
997	xxxxxxxx
998	xxxxxxxx
999	xxxxxxxx
1000	xxxxxxxx
1001	00000100
1002	01001000
1003	00000010
1004	00000005
1005	00100000
1006	00000020
1007	00000100
1008	00000004
1009	00000002
1010	000a0000

آدرس های ۱۰۰۰ تا ۱۰۰۹ (داده ها)

NewMemory.txt

1995	xxxxxxxx
1996	xxxxxxxx
1997	xxxxxxxx
1998	xxxxxxxx
1999	xxxxxxxx
2000	xxxxxxxx
2001	xxxxxxxx
2002	xxxxxxxx
2003	xxxxxxxx
2004	00000002
2005	xxxxxxxx
2006	xxxxxxxx
2007	xxxxxxxx
2008	00000008
2009	xxxxxxxx
2010	xxxxxxxx
2011	xxxxxxxx

آدرس های ۲۰۰۰ و ۲۰۰۴ (نتیجه)

```
# Loading work.Controller
VSIM 2> run -all
# Minimum Value : Mem[2000] = 2
# Index : Mem[2004] = 8
```

Test Case #2

Instructions.txt | Memory.txt

994	xxxxxxxx
995	xxxxxxxx
996	xxxxxxxx
997	xxxxxxxx
998	xxxxxxxx
999	xxxxxxxx
1000	xxxxxxxx
1001	00000100
1002	01001000
1003	00000010
1004	90000002
1005	00100000
1006	00000020
1007	00000100
1008	00000004
1009	90000005
1010	000a0000

آدرس های ۱۰۰۰ تا ۱۰۰۹ (داده ها)

NewMemory.txt

1994	xxxxxxxx
1995	xxxxxxxx
1996	xxxxxxxx
1997	xxxxxxxx
1998	xxxxxxxx
1999	xxxxxxxx
2000	xxxxxxxx
2001	xxxxxxxx
2002	xxxxxxxx
2003	xxxxxxxx
2004	90000002
2005	xxxxxxxx
2006	xxxxxxxx
2007	xxxxxxxx
2008	00000003
2009	xxxxxxxx
2010	xxxxxxxx
2011	xxxxxxxx

آدرس های ۲۰۰۰ و ۲۰۰۴ (نتیجه)