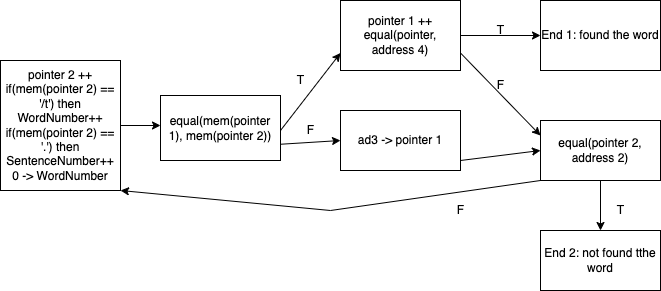
**Program Structure**

Use trap 1 to read 6 sentences

Use trap 2 to read target word

Then use loops to compare



**Memory Address**

**reserved location**

**0000 00C8**

1. Main Instructions (start from 0) (When the program is loaded to the memory it actually will not use the 0-7 locations because all the program accessing in the java code will be added 8)

2. Branches (start from 100)

3. Trap table(200 to 215)

Trap 1 300

Trap 2 400

1. Trap 1 (start from 300)

5. Trap 2 (start from 400)

1. Data Area

|  |  |  |
| --- | --- | --- |
| Address | Data | Comment |
| 1000 | Address1(1100) | start of 6 sentences |
| 1001 | Address2 | end of 6 sentences |
| 1002 | ‘.’ |  |
| 1003 | ‘/t’ |  |
| 1004 | Store Sentence (6) | counter used when store 6sentences |
| 1005 | Address3(2000) | start of target word |
| 1006 | Address4 | end of target word |
| 1007 | Index1(200) | change the index register(Trap table) |
| 1008 | pointer 1(2000) | pointer in 6 sentences |
| 1009 | pointer 2(1099) | pointer in target word |
| 1010 | Word number counter(1) |  |
| 1011 | Sentence number counter(1) |  |
| 1012 | 0 | matching failed |
| 1013 | 1 | matching succeed |

7. 6 sentences (1100 to Address 2)

8. Target word (2000 to Address 4)

**Trap1**

**Read 6 sentences**

Index 1 is 0, Index 2 is 100, Index 3 is 1000 now

**The detail of instructions:**

Opcode GPR IX I address

000000 00 00 0 00000

300)

LDX mem(1007) to x1 // change the index register 1 to 200

100001 01 11 0 00111

1000 0101 1100 0111

85C7

301)

LDX mem(200) to x1 // change the index register 1 to 300

100001 01 01 0 00000

1000 0101 0100 0000

8540

302)

LDR mem(1002) to r(0) // load ‘.’

000001 00 11 0 00010

0000 0100 1100 0010

04C2

303)

LDR mem(1001) to r(2) // load address2

000001 10 11 0 00001

0000 0110 1100 0001

06C1

304)

LDR mem(1004) to r(3) // load Sentence number

000001 11 11 0 00100

0000 0111 1100 0100

07C4

305)

IN devid(0) to r(1)

110001 01 000 00000

1100 0101 0000 0000

C500

306)

STR r(1) to mem(mem(1001)) // store one letter

000010 01 11 1 00001

0000 1001 1110 0001

09E1

307)

AIR 1 to r(2) // add 1 to address 2

000110 10 00 0 00001

0001 1010 0000 0001

1A01

308)

STR r(2) to mem(1001)

000010 10 11 0 00001

0000 1010 1100 0001

0AC1

309)

TRR r(0) to r(1)

010010 00 01 0 00000

0100 1000 0100 0000

4840

310)

JCC cc(3) to 312.

001010 11 01 0 01100

0010 1011 0100 1100

2B4C

311)

JMA to 305

001011 00 01 0 00101

0010 1100 0100 0101

2C45

312)

SOB r(3) to 305 // sentences counter

001110 11 01 0 00101

0011 1011 0100 0101

3B45

313)

JMA to 1 // all 6 sentence done.

001011 00 00 0 00001

0010 1100 0000 0001

2C01

**Trap2**

Read 1 word

400)

LDX mem(1007) to x1 // change the index register 1 to 200

100001 01 11 0 00111

1000 0101 1100 0111

85C7

401)

LDX mem(201) to x1 // change the index register 1 to 400

100001 01 01 0 00001

1000 0101 0100 0001

8541

402)

LDR mem(1003) to r(0) // load /t

000001 00 11 0 00011

0000 0100 1100 0011

04C3

403)

LDR mem(1006) to r(2) // load address4

000001 10 11 0 00110

0000 0110 1100 0110

6C6

404)

IN devid(0) to r(1)

110001 01 000 00000

1100 0101 0000 0000

C500

405)

STR r(1) to mem(mem(1006))

000010 01 11 1 00110

0000 1001 1110 0110

09E6

406)

AIR 1 to r(2)

000110 10 00 0 00001

0001 1010 0000 0001

1A01

407)

STR r(2) to mem(1006)

000010 10 11 0 00110

0000 1010 1100 0110

0AC6

408)

TRR r(0) to r(1)

010010 00 01 0 00000

0100 1000 0100 0000

4840

409)

JCC cc(3) to 2

001010 11 00 0 00010

0010 1011 0000 0010

2B02

410)

JMA to 404

001011 00 01 0 00100

0010 1100 0100 0100

2C44

**Find the match**

0)

Trap 1

011000 00 00 0 00000

0110

6000

1)

Trap 2

6001

2)

LDR mem(1003) to r(2) // load ‘/t’

000001 10 11 0 00011

0000 0110 1100 0011

06C3

3)

LDR mem(1002) to r(3) // load ‘.’

000001 11 11 0 00010

0000 0111 1100 0010

07C2

4)

LDR mem(1009) to r(0) // load pointer 2

000001 00 11 0 01001

0000 0100 1100 1001

04C9

5)

AIR 1 to r(0) // pointer 2 += 1

000110 00 00 0 00001

0001 1000 0000 0001

1801

6)

STR r(0) to mem(1009) // store pointer 2

000010 00 11 0 01001

0000 1000 1100 1001

08C9

7)

LDR mem(mem(1009)) to r(1) // load mem(pointer 2)

000001 01 11 1 01001

0000 0101 1110 1001

05E9

8)

TRR r(1) to r(2) // mem(pointer 2) == ‘/t’

010010 01 10 0 00000

0100 1001 1000 0000

4980

9)

JCC to 100 // branch 1: word number ++

001010 11 10 0 00000

0010 1011 1000 0000

2B80

10)

TRR r(1) to r(3) // mem(pointer 2) == ‘.’

010010 01 11 0 00000

0100 1001 1100 0000

49C0

11)

JCC to 104 // branch 2: sentence number ++, word number = 0

001010 11 10 0 00100

0010 1011 1000 0100

2B84

12)

LDR mem(mem(1008)) to r(0) // load mem(pointer 1)

000001 00 11 1 01000

0000 0100 1110 1000

04E8

13)

LDR mem(mem(1009)) to r(1) // load mem(pointer 2)

000001 01 11 1 01001

0000010111101001

05E9

14)

TRR r(0) to r(1) // mem(pointer 1) == mem(pointer 2)

010010 00 01 0 00000

0100 1000 0100 0000

4840

15)

JCC to 23

001010 11 00 0 10111

0010 1011 0001 0111

2B17

16)

LDR mem(1005) to r(0) // load address 3

000001 00 11 0 00101

0000010011000101

04C5

17)

STR r(0) to mem(1008) // set pointer 1 to address 3

000010 00 11 0 01000

0000100011001000

08C8

18)

LDR mem(1009) to r(0) // load pointer 2

000001 00 11 0 01001

0000010011001001

04C9

19)

LDR mem(1001) to r(1) // load address 2

000001 01 11 0 00001

0000010111000001

05C1

20)

TRR r(0) to r(1) // pointer 2 == address 2

010010 00 01 0 00000

0100100001000000

4840

21)

JCC to 117// branch 4: end 2

001010 11 10 0 10001

0010 1011 1001 0001

2B91

22)

JMA to 4 // loop

001011 00 00 0 00100

0010 1100 0000 0100

2C04

23)

LDR mem(1008) to r(0) // load pointer 1

000001 00 11 0 01000

0000010011001000

04C8

24)

AIR 1 to r(0) // pointer 1 += 1

000110 00 00 0 00001

0001100000100001

1821

25)

STR r(0) to mem(1008) // store pointer 1

000010 00 11 0 01000

0000100011001000

08C8

26)

LDR mem(1006) to r(1) // load address 4

000001 01 11 0 00110

0000010111000110

05C6

27)

TRR r(0) to r(1) // pointer 1 == address 4

010010 00 01 0 00000

0100100001000000

4840

28)

JCC to 110// branch 3 : end 1

001010 11 10 0 01010

0010 1011 1000 1010

2B8A

29)

JMA to 18

001011 00 00 0 10010

0010 1100 0001 0010

2C12

**Branches**

100)

LDR mem(1010) to r(1) // load word number

000001 01 11 0 01010

0000010111001010

05CA

101)

AIR 1 to r(1) // word number += 1

000110 01 00 0 00001

0001100100000001

1901

102)

STR r(1) to mem(1010) // store word number

000010 01 11 0 01010

0000 1001 1100 1010

09CA

103)

JMA to 10

001011 00 00 0 01010

0010 1100 0000 1010

2C0A

104)

LDR mem(1011) to r(1) // load sentence number

000001 01 11 0 01011

0000010111001011

05CB

105)

AIR 1 to r(1) // sentence number += 1

000110 01 00 0 00001

0001100100000001

1901

106)

STR r(1) to mem(1011) // store sentence number

000010 01 11 0 01011

0000100111001011

09CB

107)

LDR mem(1013) to r(1) // load 1

000001 01 11 0 01101

0000010111001101

05CD

108)

STR r(1) to mem(1010) // set word number to 1

000010 01 11 0 01010

0000100111001010

09CA

109)

JMA to 12

001011 00 00 0 01100

0010 1100 0000 1100

2C0A

110)

LDR mem (1013) to r(0) // load 1

000001 00 11 0 01101

0000 0100 1100 1101

04CD

111)

LDR mem(1011) to r(1) // load sentence number

000001 01 11 0 01011

0000010111001011

05CB

112)

LDR mem(1010) to r(2) // load word number

000001 10 11 0 01010

0000011011001010

06CA

113)

OUT r(0) to device(1)

110010 00 00 0 00001

1100100000000001

C801

114)

OUT r(1) to device(1)

110010 01 00 0 00001

1100100100000001

C901

115)

OUT r(2) to device(1)

110010 10 00 0 00001

1100101000000001

CA01

116)

0

117)

LDR mem (1012) to r(0) // load 0

000001 00 11 0 01100

0000010011001100

04CC

118)

OUT r(0) to divice(1)

110010 00 00 0 00001

1100100000000001

C801

119)

0