دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی برق

مقطع
کارشناسی

عنوان
مسیریابی

درس
معماری کامپیوتر

نگارش
زهرا عربی
9523083

استاد
دکتر راعی

دی ماه 1397

1- الگوریتم:

در این کد به گونه ای میتوان گفت که از روش بازگشتی استفاده شده بدین صورت که مختصات ابتدایی و انتهایی در هزارتو را مشخص میکنیم.
از نقطه ی شروع،شروع به حرکت میکنیم و با هر بار جا به جایی مختصات نقطه ی جدید را با نقطه ی انتهایی مقایسه میکنیم و اگر برابر نباشد حرکت را ادامه میدهیم و این کار را تا انجایی تکرار میکنیم که به نقطه ی انتهایی برسیم و پس از ان برنامه متوقف میشود.
4 نوع حرکت داریم:
پایین:شماره ی سطر یکی اضافه میشود ولی شماره ی ستون ثابت میماند.
بالا:شماره ی سطر یکی کم میشود ولی شماره ی ستون ثابت میماند.
راست:شماره ی سطر تغییری نمیکند ولی شماره ی ستون یکی اضافه میشود.
چپ:شماره ی سطر تغییری نمیکند ولی شماره ی ستون یکی کم میشود.
از نقطه ی شروع و سپس در ادامه برای هر نقطه 4 حرکت را چک میکنیم در صورتی که مختصات جدید به عدد 1 ختم شود،مجاز به حرکت است در غیر اینصورت مجاز به حرکت نیست.(روی مسیر آمده نیز نمیتواند بازگشتی داشته باشد.)
ترتیب حرکت ها و اولویت ان ها برای این کد مهم نیست چون برای هر نقطه ی جدید هر 4 حرکت باید چک بشودو از بین 4 حرکت فقط مجاز به انتخاب یکی می باشد.
در صورتیکه مسیری از نقطه ی ابتدا به انتها وجود نداشته باشد؛کد الگوریتم را دنبال کرده تا به نقطه ی بن بست برسد و سپس برنامه متوقف می شود.

2- توضیح کد:

```
ldr     r0,rownum       ; r0 := M[rownum] = 8
adr     r1,rows         ; r1 := rows
```

ابتدا مرتبه ی ماتریس که با rownum مشخص میشود را در رجیستر ro میریزیم.سپس آدرس متغیری که حاوی آدرس های اول هر سطر است را در r1 میریزیم.

```
sub     r6,r0,#1
add     r4,r1,r6,lsl #2   ;r2,r3,r4,r5 all address
ldr     r5,[r4]
sub     r6,r6,#1
add     r5,r5,r6
```

در اینجا میخواهیم ادرس انتهایی مسیر را بیابیم.طبق فرمول گفته شده در سوال پایان همیشه در خانه ی a(a-1) -1 قرار دارد.پس برای محاسبه ی آدرس سطر و ستون چنین میکنیم:

(1 - مرتبه ی ماتریس) *4 + آدرس سطر اول = آدرس سطر خانه ی پایان
(2 – مرتبه ی ماتریس) + آدرس ستون اول =آدرس ستون خانه ی پایان

بدین ترتیب آدرس ها را محاسبه میکنیم و سطر خانه ی پایان را در r4 و ستون خانه ی پایان را در r5 میریزیم.

```
add     r2,r1,#+4
ldr     r3,[r2]
add     r3,r3,#+1
```

در اینجا آدرس خانه ی شروع را به دست می آوریم.که همیشه خانه ی شروع در سطر دوم (چون آدرس سطر ها کلمه هستند 32 بیت 32 بیت یعنی 4 بایت 4 بایت اضافه می شوند.) و ستون دوم (ادرس ستون ها ا بایتی هستند.) قرار دارد.آدرس سطر را در رجیستر r2 و آدرس ستون را در رجیستر r3 قرار میدهیم و محتویات خانه ی ماتریس همان محتویات رجیستر r3 می باشد.

```
Add   r10,r0,#+2
```

اینجا شمارشگری میسازیم تا خانه های ماتریس را به صورت دسیمال بشمارد و در تابع پرینت این رجیستر را در مغیر prout برای نمایش میریزیم.
مقدار اولیه ی این متغیر (مرتبه ی ماتریس + 2) میباشد.
اگر حرکت رو به بالا یا پایین داشته باشیم از مقدار آن به اندازه ی مرتبه ی ماتریس کم یا به مقدار آن به اندازه ی مرتبه ی ماتریس اضافه میشود.
اگر حرکت به سمت راست یا چپ داشته باشیم به مقدار یکی اضافه یا کم میشود.

```
LDR  r8,proute
```

محتوای متغیر prout را در r8 قرار میدهیم.prout یک اشاره گر به rout متغیر نوشتنی ما میباشد.
محتوای prout آدرس rout میشود.

```
mov  r11,#0
```

به رجیستر r11 مقدار اولیه صفر میدهیم.این رجیستر برای ثبت جهت حرکت قبلی استفاده میشود برای اینکه رو مسیر قبلی باز نگردیم.
برای مثال هنگامی که هر کدام از حرکت ها انجام شدند این رجیستر مقدار به خصوصی میگیرد.(بعد از حرکت پایین 1 میشود.بعد ازحرکت راست 2 میشود.بعد از حرکت بالا 3 میشود و بعد از حرکت چپ 4 میشود.)
برای هر حرکت ابتدا باید این رجیستر چک بشود بدین صورت که اگر حرکت قبلی در جهت پایین بوده باشد نمیتواند حرکت بالا داشته باشد یا اگر حرکت قبلی در جهت راست بوده باشد نمیتواند حرکت چپ داشته باشد.

```
ldr    r6,[r3]
cmp    r6,#1
bl     print
bal    endpoint
```

در این مرحله محتوای خانه ی اول را در رجیستر r6 میریزیم و آن را با یک مقایسه میکنیم اگر مساوی بود و اگر مساوی بود وارد تابع پرینت شود.(که البته میدانیم همواره این شرط برقرار است زیرا خانه ی شروع باید یک باشد) و بعد از آن باید به endpoint بپرد.
با دستور bal فضایی قبل از endpoint به وجود می آوریم که توابع دلخواه را که نمیخواهیم اول برنامه اجرا شوند و فقط در صورتی که صدا زده شوند اجرا شوند را قرار دهیم.

```
print    STR  r10,[r8]      ; M[route] := r2 = 11
         add  r8,r8,#+4
         mov  r15,r14
```

محتوای rout مقدار رجیستر r10 میشود.سپس آدرس rout یا r8 را به اندازه ی یک کلمه(4 بایت) افزایش میدهیم.که عدد بعدی در خانه ی حافظه ی بعدی چاپ شود.
سپس با دستور mov ، r14 که دارای آدرس برگشت است را در pc قرار میدهیم تا از تابع(ساب روتین) خارج شده و به ادامه ی کد اصلی برگردیم.

```
endpoint  cmp   r2,r4
          bhi   endloop
          bne   down
          cmp   r3,r5
          bhi   endloop
```

```
        bne     down
        bl      print
        bal     endloop
```

نقطه ی endpointنقطه ی بازگشتی کد میباشد در آدرس سطر نقصه ی فعلی با نقطه ی انتهایی چک میشود و سپس آدرس ستون آن با نقطه ی انتهایی چک میشود.اگر مختصات نقطه ی فعلی بزرگتر از نقطه ی انتهایی باشد باید برنامه تمام شود در غیر اینصورت اگر مختصات ها برابر نباشد باید وارد دستور حرکت ها بشویم و اگر هیچ کدام از شرط های بالا نبود یعنی مختصات موجود با مختصات نهایی یکی شده در این صورت باید وارد تابع پرینت شود و بعد از آن برنامه باید تمام شود.

```
down    cmp   r11,#+3
        beq    right
        add   r7,r2,#+4
        ldrb   r9,[r3,r0]
        cmp    r9,#1
        bne   right
        mov    r2,r7
        add    r3,r3,r0
        add   r10,r10,r0
        mov   r11,#1
        bl      print
```

اولین حرکت را پایین در نظر گرفته شده.ابتدا باید چک بشود که حرکت رو به سمت بالا نباشد(اگر جرکت قبلی رو به سمت بالا باشد یعنی 3=(r11) در اینصورت این حرکت مجاز نیست و باید به حرکت بعدی برود.
شماره ی سطر آن باید 4 بایت اضافه شود و متناسب با آن شماره ی ستون آن نیز اضافه شود تا بتوان محتوای آن را چک کرد اگر 1 بود شماره ی سطر و ستون تغییر پیدا میکند(حرکت کرده) شمارنده با توجه یه نوع حرکت اضافه میشود و r11 با توجه به نوع حرکت مقدار مخصوص آن را میگیرد و سپس باید وارد تابع پرینت شود تا شماره ی خانه ی جدید را چاپ کند.

```
right   cmp   r11,#4
        beq    up
        add   r7,r3,#1
        ldrb   r9,[r7]
        cmp   r9,#1
        bne   up
        mov   r3,r7
        add   r10,r10,#1
        mov    r11,#2
        bl      print
```

حرکت سمت راست همانند حرکت پایین است با این تبرای حرکت سمت راست باید چک بشود که حرکت قبلی چپ بوده است یا خیر.اگر چپ باشد مجاز به حرکت راست نیستیم.
در حرکت راست فقط شماره ی ستون عوض میشود و شماره ی سطر ثابت باقی می ماند.
و بقیه ی مراحل و توضیحات همانند حرکت پایین است.

```
up      cmp   r11,#1
        beq    left
        sub    r7,r2,#4
        sub    r12,r3,r0
        ldrb   r9,[r12]
        cmp    r9,#1
        bne    left
        mov    r2,r7
        mov    r3,r12
        sub    r10,r10,r0
        mov    r11,#3
        bl     print
```

در حرکت بالا حرکت قبلی نباید پایین باشد.برای حرکت بالا شماره ی سطر چهار بایت کم میشود و شماره ی ستون متناسب با آن کم میشود.و این کم کردن و چک کردن در یک مرحله با توجه به ارور های برنامه انجام پذیر نبود بنابراین در دو مرحله (ابتدا کم میکنیم و سپس مقدار آن را چک میکنیم)نوشته شد.
بقیه ی توضیحات مشابه حرکت های قبلی است.

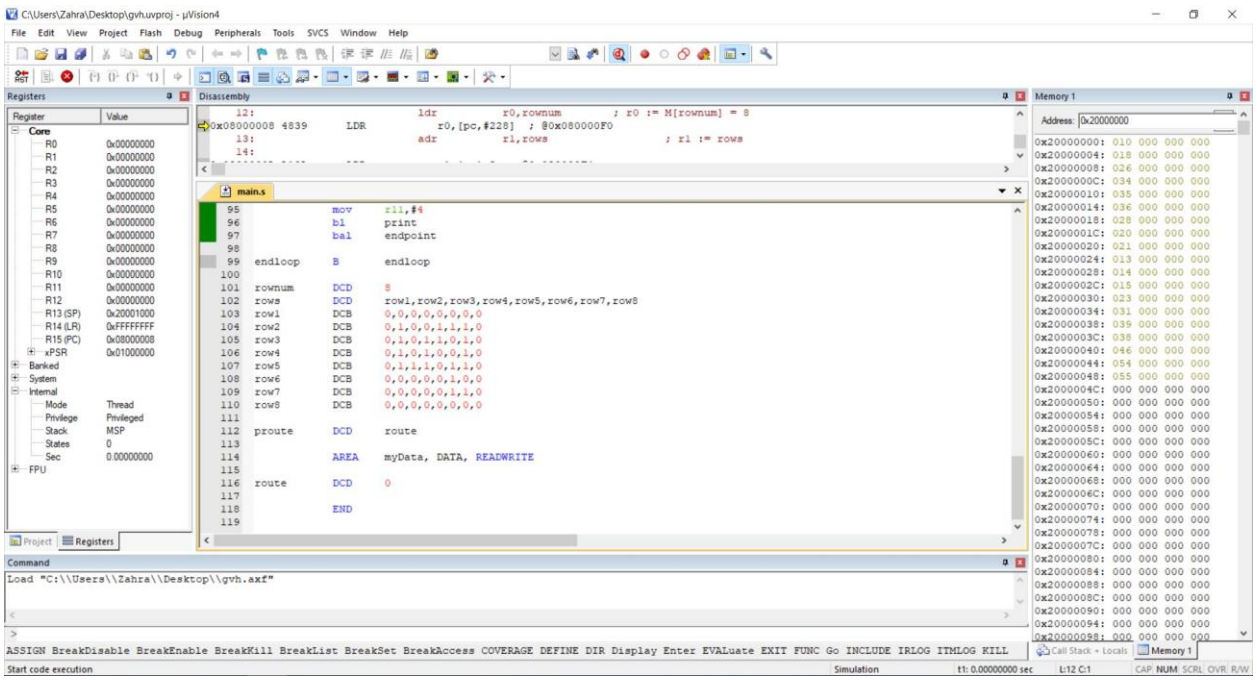
```
left    cmp   r11,#2
        beq   endpoint
        sub   r7,r3,#1
        ldrb  r9,[r7]
        cmp   r9,#1
        bne    endpoint
        mov   r3,r7
        sub   r10,r10,#1
        mov   r11,#4
        bl    print
        bal   endpoint

endloop         B       endloop
```

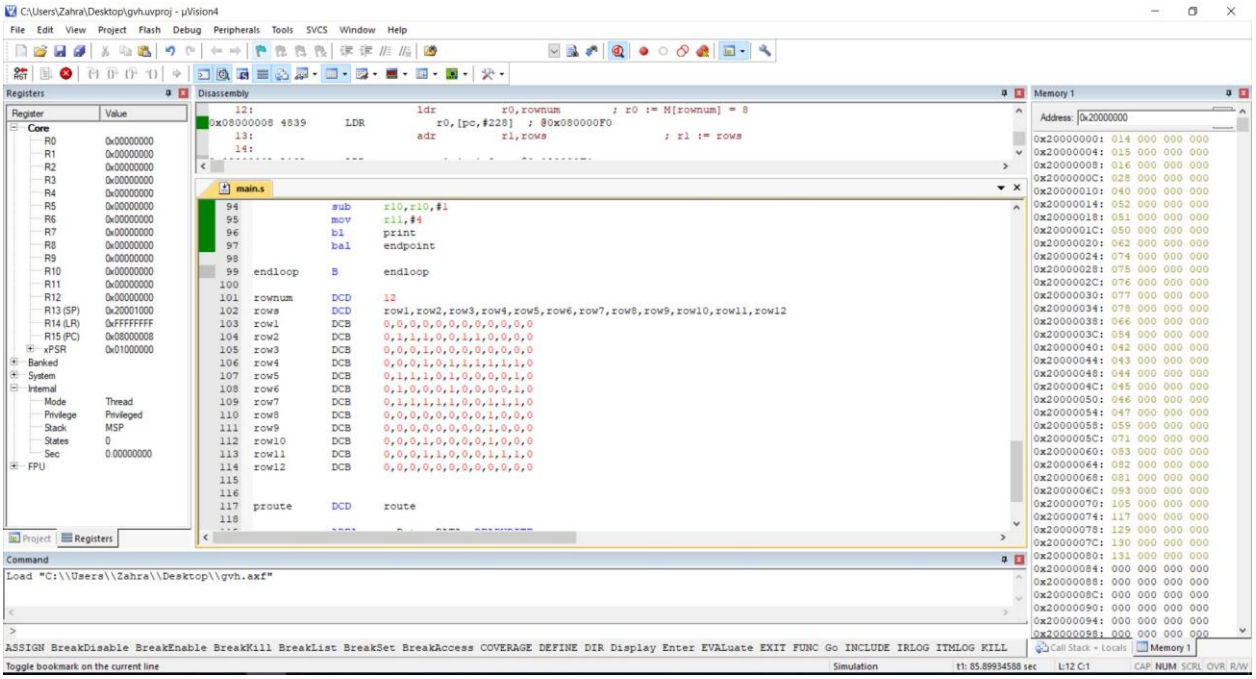
حرکت سمت چپ زمانی مجاز است که حرکت قبلی راست نباشد.

در حرکت سمت چپ شماره ی سطر تغییر نمیکند و فقط شماره ی ستون یکی کم میشود که همانند حرکت بالا کم کردن و چک کردن در دو مرحله صورت گرفته است.بقیه ی توضیحات شبیه حرکت های بالایی است.
و در اخر حلقه ی بی نهایت endloop را داریم که برای اتمام برنامه ها به کار برده میشود.

3- تصاویری از نتایج:



ماتریس 8*8


C:\Users\Zahra\Desktop\gvh.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Register Value
Core
R0 0x00000000
R1 0x00000000
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20001000
R14 (LR) 0xFFFFFFFF
R15 (PC) 0x08000008
xPSR 0x01000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 0
Sec 0.00000000
FPU
Disassembly
12: ldr r0,rownum ; r0 := M[rownum] = 8
0x08000008 4839 LDR r0,[pc,#228] ; @0x080000F0
13: adr r1,rows ; r1 := rows
14:
main.s
94 sub r10,r10,#1
95 mov r11,#4
96 bl print
97 bal endpoint
98
99 endloop B endloop
100
101 rownum DCD 12
102 rows DCD row1,row2,row3,row4,row5,row6,row7,row8,row9,row10,row11,row12
103 row1 DCB 0,0,0,0,0,0,0,0,0,0,0,0
104 row2 DCB 0,1,1,1,0,0,1,1,0,0,0,0
105 row3 DCB 0,0,0,1,0,0,0,0,0,0,0,0
106 row4 DCB 0,0,0,1,0,1,1,1,1,1,1,0
107 row5 DCB 0,1,1,1,0,1,0,0,0,0,1,0
108 row6 DCB 0,1,0,0,0,1,0,0,0,0,1,0
109 row7 DCB 0,1,1,1,1,1,0,0,1,1,1,0
110 row8 DCB 0,0,0,0,0,0,0,0,1,0,0,0
111 row9 DCB 0,0,0,0,0,0,0,0,1,0,0,0
112 row10 DCB 0,0,0,1,0,0,0,0,1,0,0,0
113 row11 DCB 0,0,0,1,1,0,0,0,1,1,1,0
114 row12 DCB 0,0,0,0,0,0,0,0,0,0,0,0
115
116
117 proute DCD route
118
Memory 1
Address: 0x20000000
0x20000000: 014 000 000 000
0x20000004: 015 000 000 000
0x20000008: 016 000 000 000
0x2000000C: 028 000 000 000
0x20000010: 040 000 000 000
0x20000014: 052 000 000 000
0x20000018: 051 000 000 000
0x2000001C: 050 000 000 000
0x20000020: 062 000 000 000
0x20000024: 074 000 000 000
0x20000028: 075 000 000 000
0x2000002C: 076 000 000 000
0x20000030: 077 000 000 000
0x20000034: 078 000 000 000
0x20000038: 066 000 000 000
0x2000003C: 054 000 000 000
0x20000040: 042 000 000 000
0x20000044: 043 000 000 000
0x20000048: 044 000 000 000
0x2000004C: 045 000 000 000
0x20000050: 046 000 000 000
0x20000054: 047 000 000 000
0x20000058: 059 000 000 000
0x2000005C: 071 000 000 000
0x20000060: 083 000 000 000
0x20000064: 082 000 000 000
0x20000068: 081 000 000 000
0x2000006C: 093 000 000 000
0x20000070: 105 000 000 000
0x20000074: 117 000 000 000
0x20000078: 129 000 000 000
0x2000007C: 130 000 000 000
0x20000080: 131 000 000 000
0x20000084: 000 000 000 000
0x20000088: 000 000 000 000
0x2000008C: 000 000 000 000
0x20000090: 000 000 000 000
0x20000094: 000 000 000 000
0x20000098: 000 000 000 000
Project Registers
Command
Load "C:\\Users\\Zahra\\Desktop\\gvh.axf"
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go INCLUDE IRLOG ITMLOG KILL
Call Stack + Locals
Memory 1
Toggle bookmark on the current line
Simulation
t1: 85.89934588 sec
L:12 C:1
CAP NUM SCRL OVR R/W

ماتریس 12*12