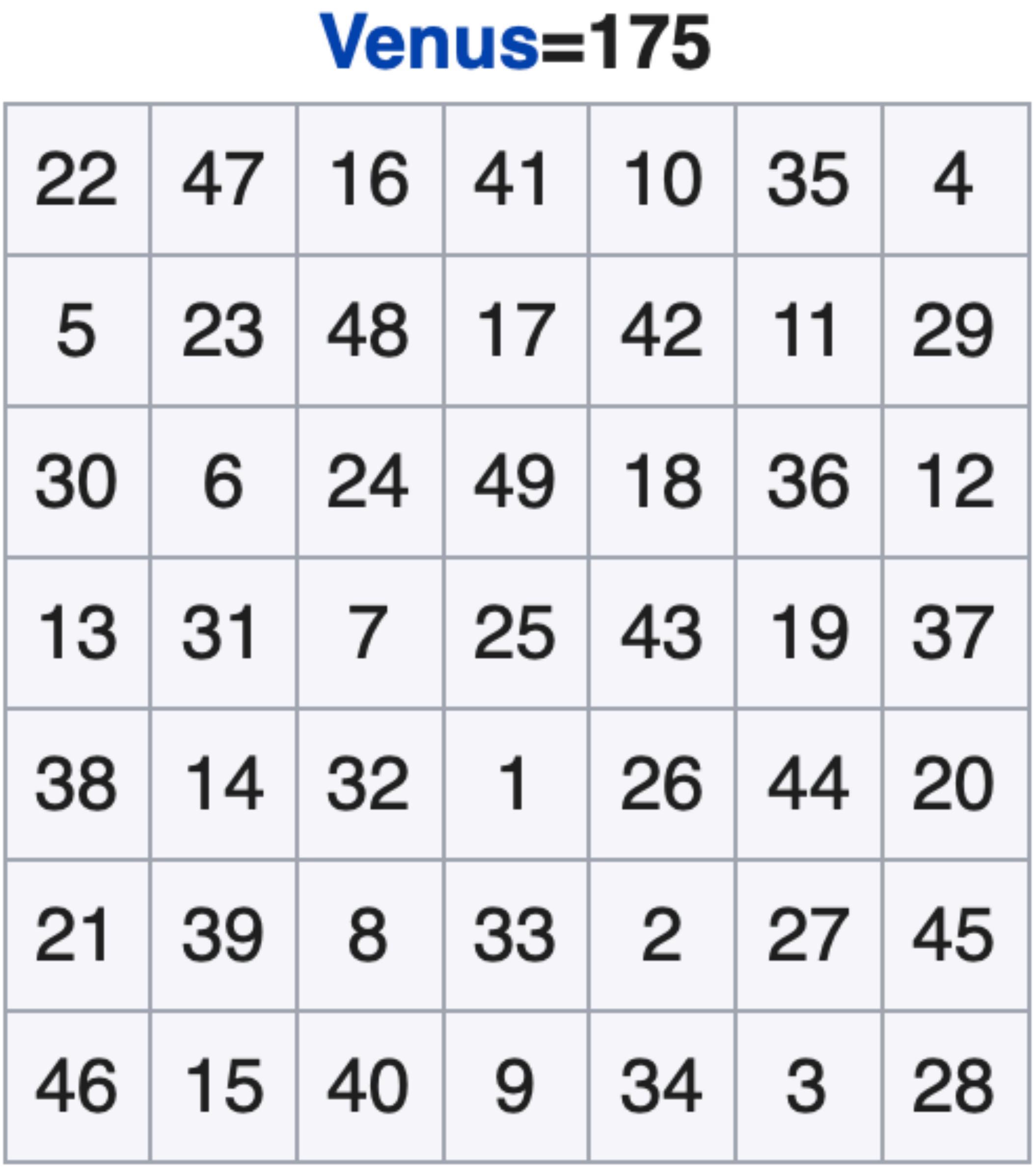
1. Magic Squares (AOC 1.3.2 execise 21 . pages 117)

A magic square of order n is an arrangement of the numbers 1 though n^2 in square array in such a way that sum of each row and column in $n(n^2+1)/2$, and so is the sum of two main diagonals . Figure 1 shows a magic sqaure of order 7.



Venus=175

22	47	16	41	10	35	4
5	23	48	17	42	11	29
30	6	24	49	18	36	12
13	31	7	25	43	19	37
38	14	32	1	26	44	20
21	39	8	33	2	27	45
46	15	40	9	34	3	28

The rules for generating it is easily seen. Start with 1 jumt below the middle square, then go down to the right diagonally - when running off the edge imagine an entire plane titled with squares - until reaching a filled square;then drop down two spaces from the most-recently-filled square and continue .

This method works whenever n is odd.

2. Solution for the problem

- the odd number of SS
- set the size of $N = n^2 = SS^2$
- set the region of the matrix $SQ[i][j]$
- location the center of the matrix $CENTER = SS/2$
- put the 1 into the position $i = CENTER + 1, j = CENTER$
- put the offset from SQ into IDX

3. Define the variables to hold the matrix.

Venus = 23^2

```
SS       EQU     23      define the martrix size ,that n is odd
     |
SQ       CON     0       ; start of the martrix in memory
      |
         ORIG    *+SS*SS ; offset the memory location to save table of data
W        CON     SS      ; the width of the martrix
      |
CENTER   CON     SS/2    ; the center of the martrix
      |
N        CON     SS*SS   ; the size of the martrix
      |
IDX      CON     0       ; the offset of martrix in the one dimsion
      |
MSG      ALF     1       ; the msg to output
      |
         ORIG    *+SS    ; the end of msg
```

4. Sub-routine to find the offset of matrix

```
INDEX   STJ     9F      ; find the index of the martrix by rI1 = i ,rI2 = j
       |
        ENTA    0       ; the size of the index continue increase , with
        ENTX    0,1     ; with the reminder of the width of matrix the the
        DIV     W       ; the postion within the martirx.
        SLAX    5
        MUL     W
        STX     IDX     ; the rI1*W
        ENTA    0
        ENTX    0,2
        DIV     W
        SLAX    5       ; the offset = rI1*W+rI2
        ADD     IDX     ; the offset of the martrix
        STA     IDX
9H      JMP     *
```

Use the $iI1 = i, iI2 = j$ store the offset of martrix into IDX ,
the tricks is the always increase $iI1, iI2$, while using the W to divide $iI1, iI2$, take the remainder alway with the matrix.

5. Main program

Alogrithm of magic square

1. [start with 1 jumt below the middle square]
2. [go down to the right diagonally] $iI1 = i + 1, iI2 = j + 1$
3. [check if reach filled square] load memory cell in offset
4. [if not filled , save sequence into the location,loop to 2]
5. [if filled , just go down $iI1 = i + 2, iI2 = j$

```
START   ENT6    1               ; start iI6 as numbers
       |mixasm /tmp/samples/square.mixal
        LD1     CENTER          ; iI1 = i , iI2 = j
        LD2     CENTER
        INC1    1               ; the first position is i+1
        JMP     INDEX
        LD5     IDX             ; find offset the martirx
        ST6     0,5             ; store the 1st into the memory
1H      INC1    1               ; normal increase i+1,j+1
       |
        INC2    1
        INC6    1
        JMP     INDEX           ; find the offset of matrix
        LD5     IDX
        LDA     SQ,5            ; test if the cell can place rI6
        JAZ     2F              ; can put the number
        INC1    1               ; can not put the number , just i+2,j
        DEC2    1
        JMP     INDEX           ; find the offset
        LD5     IDX             ;
2H      ST6     SQ,5            ; store the number.
       |
        CMP6    N               ; if the number small than the N*N,then loop
        JL      1B
        JMP     SHOW            ; show the result of the matrix
        HLT
        END     START
```

6. Show the matrix

This sub-routine parse the memory of matrix into the square numbers.

```
SHOW    STJ     9F      ; show the SQUARE in the decimal number.
      |
        ENT1    0
1H      ENT2    0       ; inner loop
      |
        ENTX    0
        LDA     SQ,1
        CHAR
        ENT3    0
        ST3     MSG,2(0:1)      ; space in the first byte of MSG
        STX     MSG,2(2:5)      ; 04d to express the decimal
        INC1    1
        INC2    1
        CMP2    W
        JL      1B+1            ; jump of the inner
        OUT     MSG(19)
        CMP1    N               ; compare the offset iI1 < N*N
        JL      1B
9H      JMP     *
```

7. Test N= 3, N=7 ,N=23

```
    SS      EQU     3       define the martrix size ,that n is odd
```

```
AppledeMBP-2:samples rqz$ mixvm -r square.mix
Program loaded. Start address: 3037
Running ...
 0004 0009 0002
 0003 0005 0007
 0008 0001 0006
... done

AppledeMBP-2:samples rqz$ mixvm -r square.mix
Program loaded. Start address: 3037
Running ...
 0022 0047 0016 0041 0010 0035 0004
 0005 0023 0048 0017 0042 0011 0029
 0030 0006 0024 0049 0018 0036 0012
 0013 0031 0007 0025 0043 0019 0037
 0038 0014 0032 0001 0026 0044 0020
 0021 0039 0008 0033 0002 0027 0045
 0046 0015 0040 0009 0034 0003 0028
... done

AppledeMBP-2:samples rqz$ mixvm -r square.mix
Program loaded. Start address: 3037
```

```
Running ...
 0254 0519 0232 0497 0210 0475 0188 0453 0166 0431 0144 0409 0122 0387
 0013 0255 0520 0233 0498 0211 0476 0189 0454 0167 0432 0145 0410 0123
 0278 0014 0256 0521 0234 0499 0212 0477 0190 0455 0168 0433 0146 0411
 0037 0279 0015 0257 0522 0235 0500 0213 0478 0191 0456 0169 0434 0147
 0302 0038 0280 0016 0258 0523 0236 0501 0214 0479 0192 0457 0170 0435
 0061 0303 0039 0281 0017 0259 0524 0237 0502 0215 0480 0193 0458 0171
 0326 0062 0304 0040 0282 0018 0260 0525 0238 0503 0216 0481 0194 0459
 0085 0327 0063 0305 0041 0283 0019 0261 0526 0239 0504 0217 0482 0195
 0350 0086 0328 0064 0306 0042 0284 0020 0262 0527 0240 0505 0218 0483
 0109 0351 0087 0329 0065 0307 0043 0285 0021 0263 0528 0241 0506 0219
 0374 0110 0352 0088 0330 0066 0308 0044 0286 0022 0264 0529 0242 0484
 0133 0375 0111 0353 0089 0331 0067 0309 0045 0287 0023 0265 0507 0243
 0398 0134 0376 0112 0354 0090 0332 0068 0310 0046 0288 0001 0266 0508
 0157 0399 0135 0377 0113 0355 0091 0333 0069 0311 0024 0289 0002 0267
 0422 0158 0400 0136 0378 0114 0356 0092 0334 0047 0312 0025 0290 0003
 0181 0423 0159 0401 0137 0379 0115 0357 0070 0335 0048 0313 0026 0291
 0446 0182 0424 0160 0402 0138 0380 0093 0358 0071 0336 0049 0314 0027
 0205 0447 0183 0425 0161 0403 0116 0381 0094 0359 0072 0337 0050 0315
 0470 0206 0448 0184 0426 0139 0404 0117 0382 0095 0360 0073 0338 0051
 0229 0471 0207 0449 0162 0427 0140 0405 0118 0383 0096 0361 0074 0339
 0494 0230 0472 0185 0450 0163 0428 0141 0406 0119 0384 0097 0362 0075
 0253 0495 0208 0473 0186 0451 0164 0429 0142 0407 0120 0385 0098 0363
 0518 0231 0496 0209 0474 0187 0452 0165 0430 0143 0408 0121 0386 0099
... done

AppledeMBP-2:samples rqz$
```

```
 MIX> load square.mix
Program loaded. Start address: 3037
MIX> run
Running ...
 0004 0009 0002
 0003 0005 0007
 0008 0001 0006
... done
Elapsed time: 931 /Total program time: 931 (Total uptime: 48842)
MIX> pmem -9
0000: + 00 00 00 00 04 (0000000004)
0001: + 00 00 00 00 09 (0000000009)
0002: + 00 00 00 00 02 (0000000002)
0003: + 00 00 00 00 03 (0000000003)
0004: + 00 00 00 00 05 (0000000005)
0005: + 00 00 00 00 07 (0000000007)
0006: + 00 00 00 00 08 (0000000008)
0007: + 00 00 00 00 01 (0000000001)
0008: + 00 00 00 00 06 (0000000006)
0009: + 00 00 00 00 00 (0000000000)
```

```
MIX> psym
MSG                    :  16
INDEX                  :  3000
N                      :  14
CENTER                 :  13
SHOW                   :  3020
SS                     :  3
L                      :  3014
W                      :  12
IDX                    :  15
START                  :  3037
SQ                     :  0
MIX> pmem 15
0015: + 00 00 00 00 01 (0000000001)
MIX> pmem 12
0012: + 00 00 00 00 03 (0000000003)
MIX> pmem 13
0013: + 00 00 00 00 01 (0000000001)
MIX> pmem 14
0014: + 00 00 00 00 09 (0000000009)
MIX> pmem 12-16
0012: + 00 00 00 00 03 (0000000003)
0013: + 00 00 00 00 01 (0000000001)
0014: + 00 00 00 00 09 (0000000009)
0015: + 00 00 00 00 01 (0000000001)
0016: + 00 30 30 30 38 (0007989158)
MIX> pall
rA: + 30 30 30 30 30 (0511305630)
rX: + 30 30 30 30 36 (0511305636)
rJ: + 47 29 (3037)
rI1: + 00 09 (0009) rI2: + 00 03 (0003)
rI3: + 00 00 (0000) rI4: + 00 00 (0000)
rI5: + 00 01 (0001) rI6: + 00 09 (0009)
Overflow: F
Cmp: E
MIX>
```