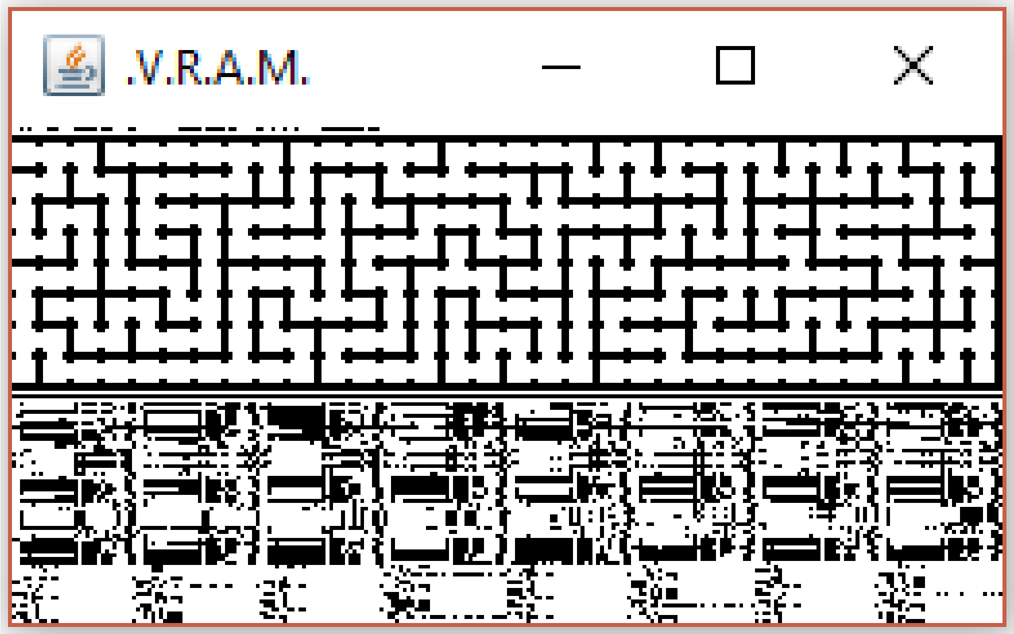
Project 1

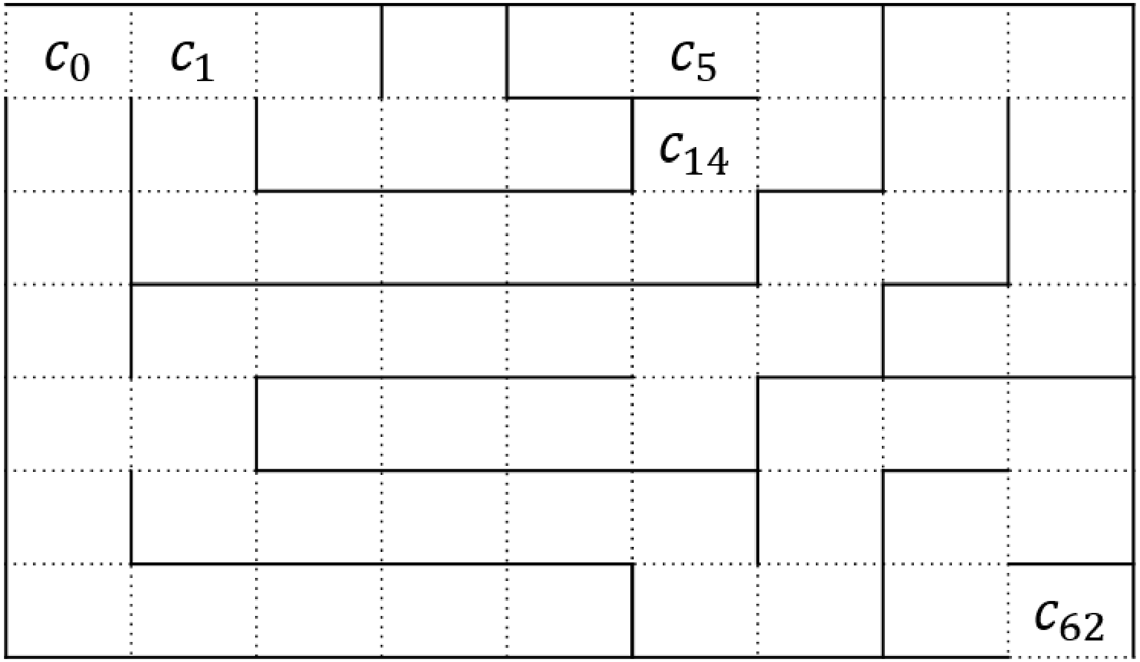
Draw a random perfect maze in the memory.



To be done by teams of **two people**

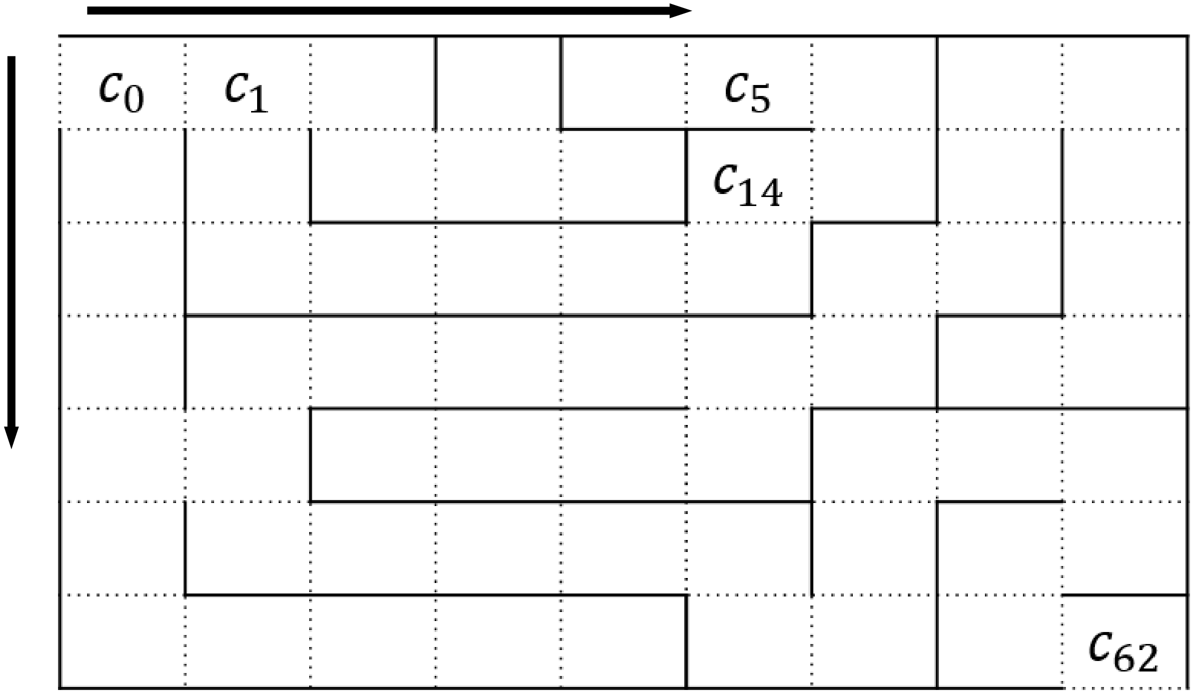
Deadline : October 29, 2017, 23:59

Perfect maze



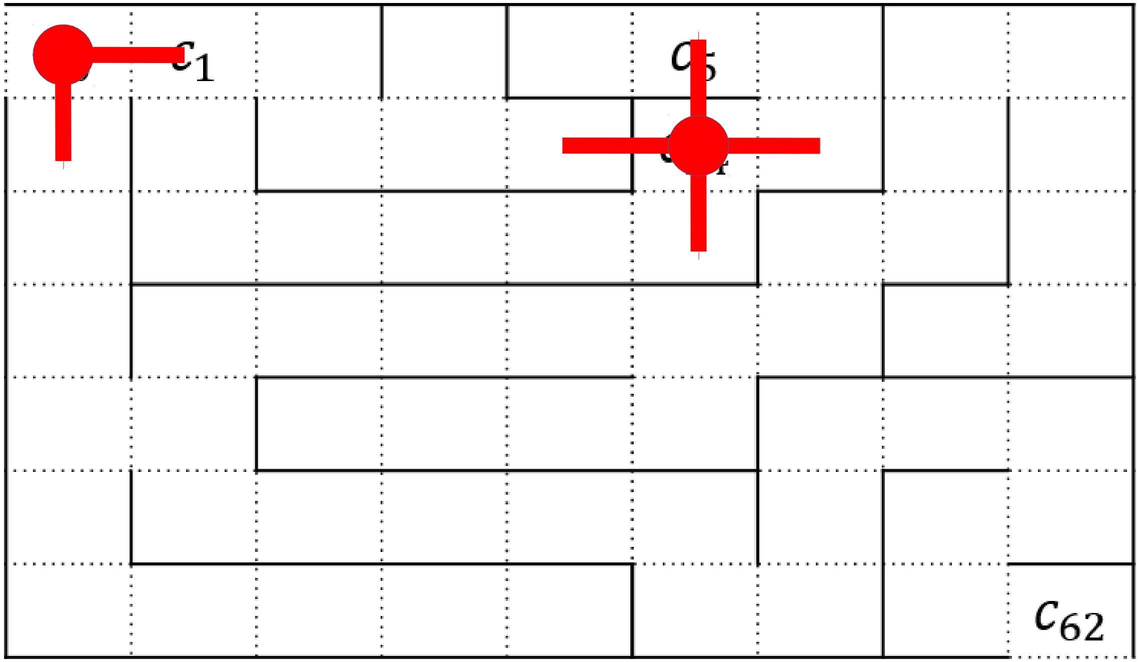
- **Perfect maze** : no cycle and a unique path between all pairs of cells

Perfect maze



- **Perfect maze** : no cycle and a unique path between all pairs of cells
- Left-to-right, top-to-bottom numbering for cells

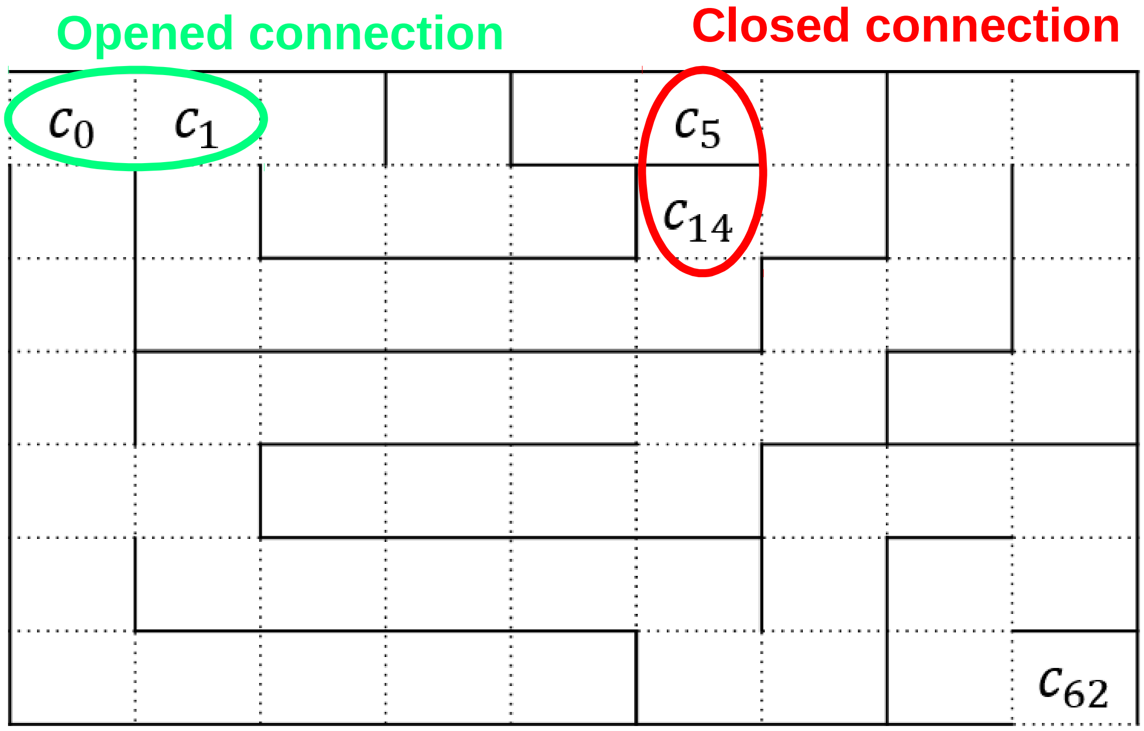
Perfect maze



Beware of borders !! (e.g. only 2 neighbours for cell c_0)

- **Perfect maze** : no cycle and a unique path between all pairs of cells
- Left-to-right, top-to-bottom numbering for cells
- A cell is connected to its 4 neighbours (top, bottom, left, right)

Perfect maze

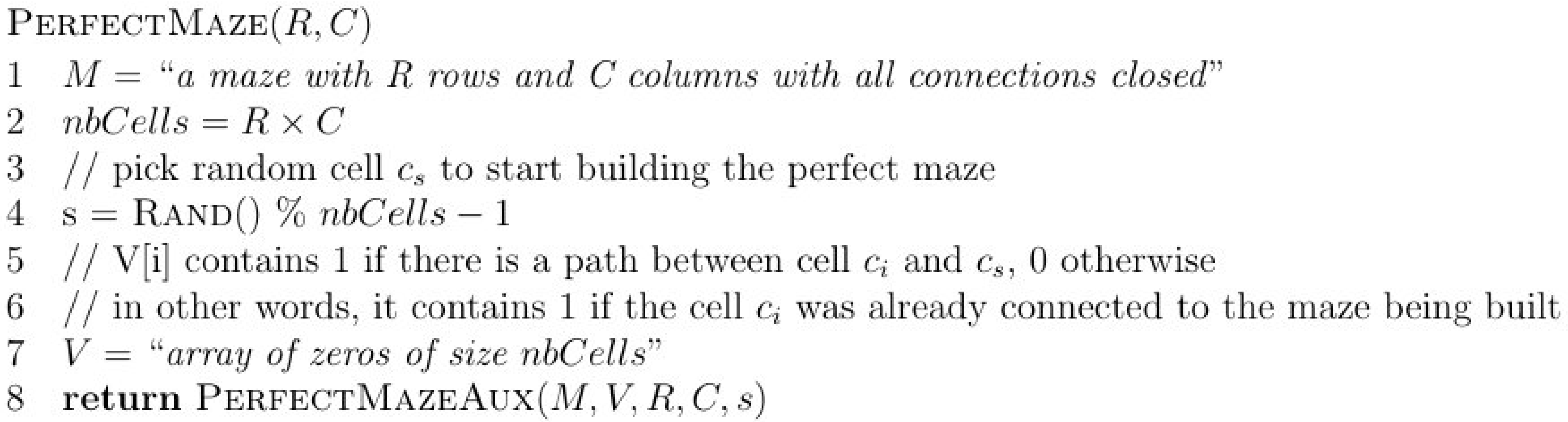


Opened connection

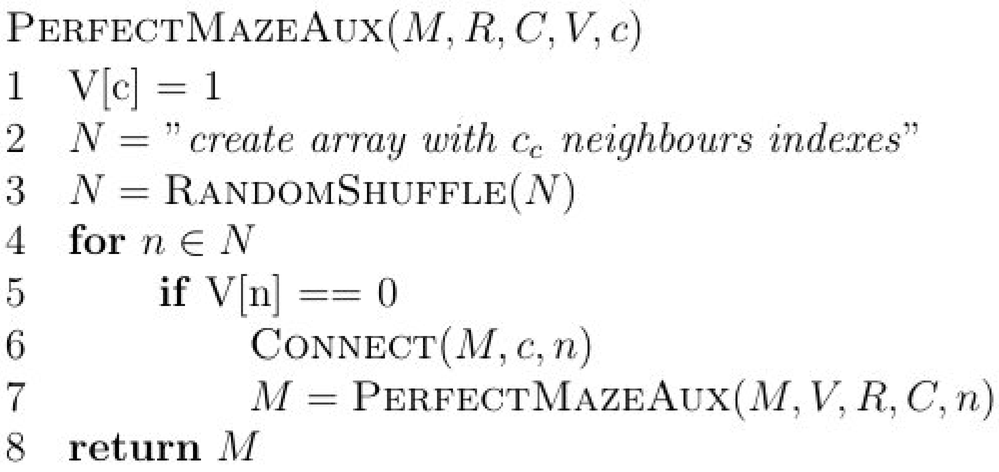
Closed connection

- **Perfect maze** : no cycle and a unique path between all pairs of cells
- Left-to-right, top-to-bottom numbering for cells
- A cell is connected to its 4 neighbours (top, bottom, left, right)
- A connection is either closed or opened

Algorithm for a perfect maze

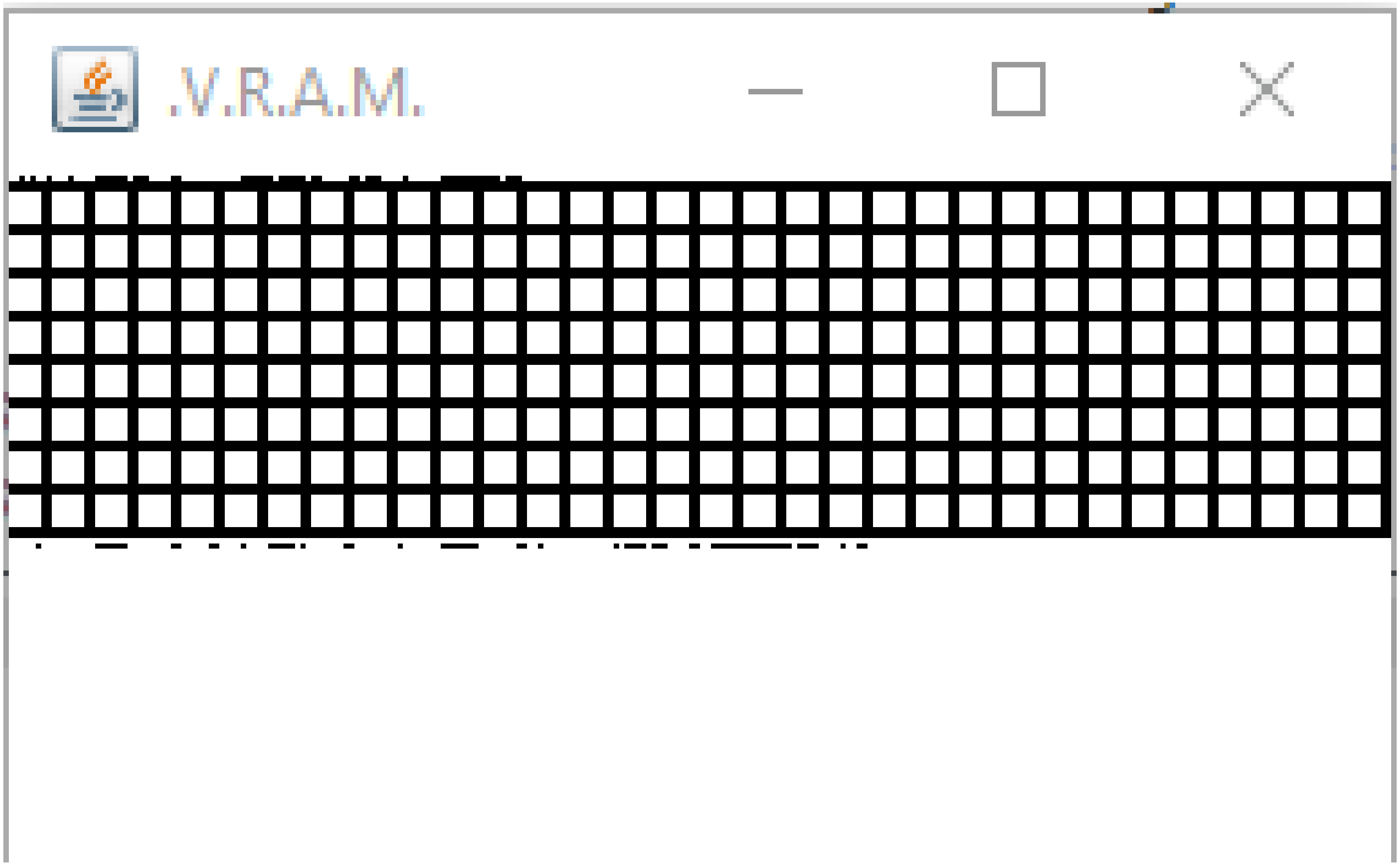


```
$\text{PerfectMaze}(R, C)$
$M$ = "a maze with $R$ rows and $C$ columns with all connections closed"
$nbCells = R \times C$
// pick random cell $c_s$ to start building the perfect maze
s = $\text{Rand}()$ % $nbCells - 1$
// V[i] contains 1 if there is a path between cell $c_i$ and $c_s$, 0 otherwise
// in other words, it contains 1 if the cell $c_i$ was already connected to the maze being built
$V$ = "array of zeros of size $nbCells$"
return $\text{PerfectMazeAux}(M, V, R, C, s)$
```



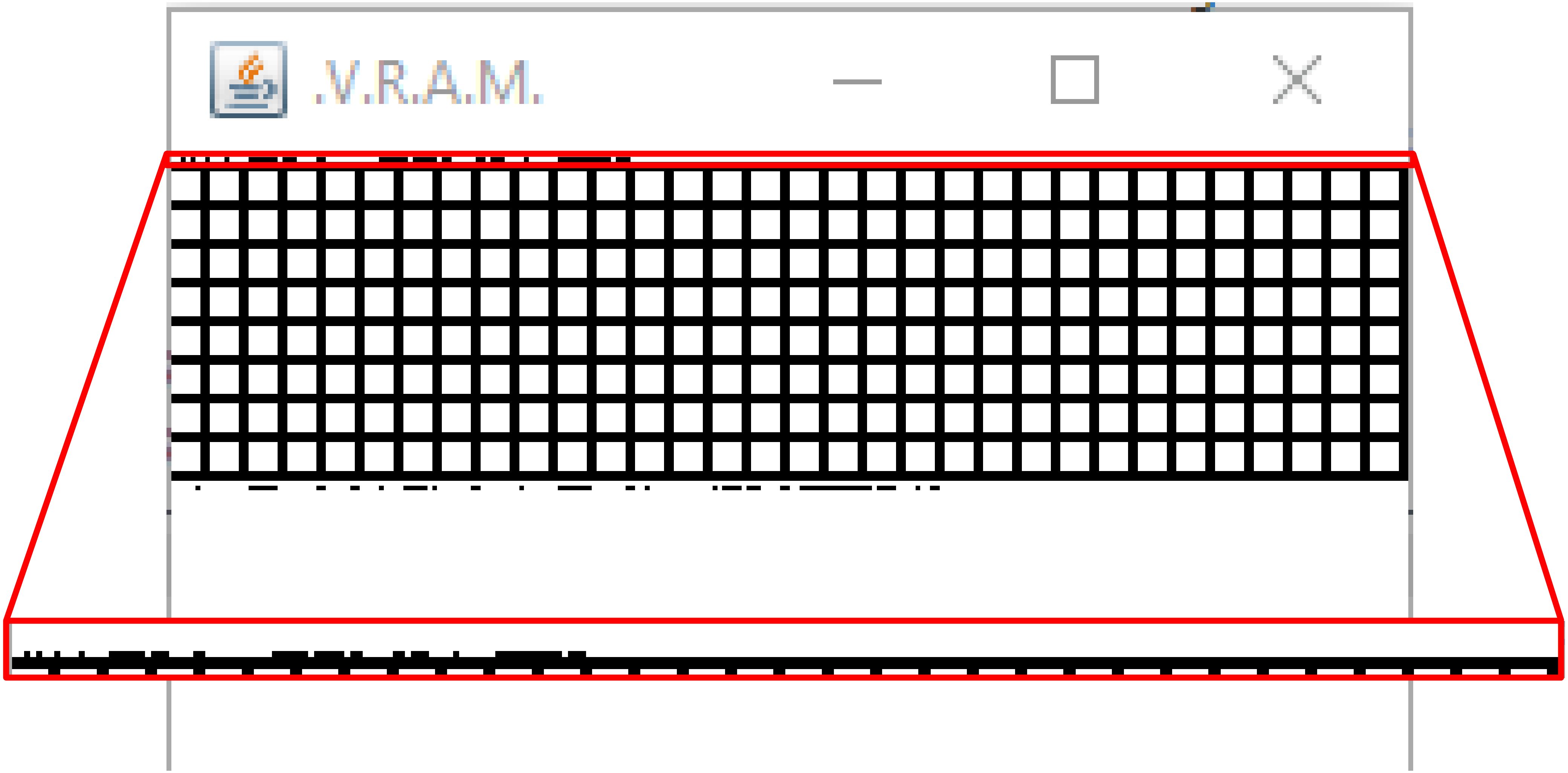
```
$\text{PerfectMazeAux}(M, R, C, V, c)$
V[c] = 1
$N$ = "create array with $c_c$ neighbours indexes"
$N = \text{RandomShuffle}(N)$
for $n \in N$
    if V[n] == 0
        $\text{Connect}(M, c, n)$
        $M = \text{PerfectMazeAux}(M, V, R, C, n)$
return $M$
```

Memory view



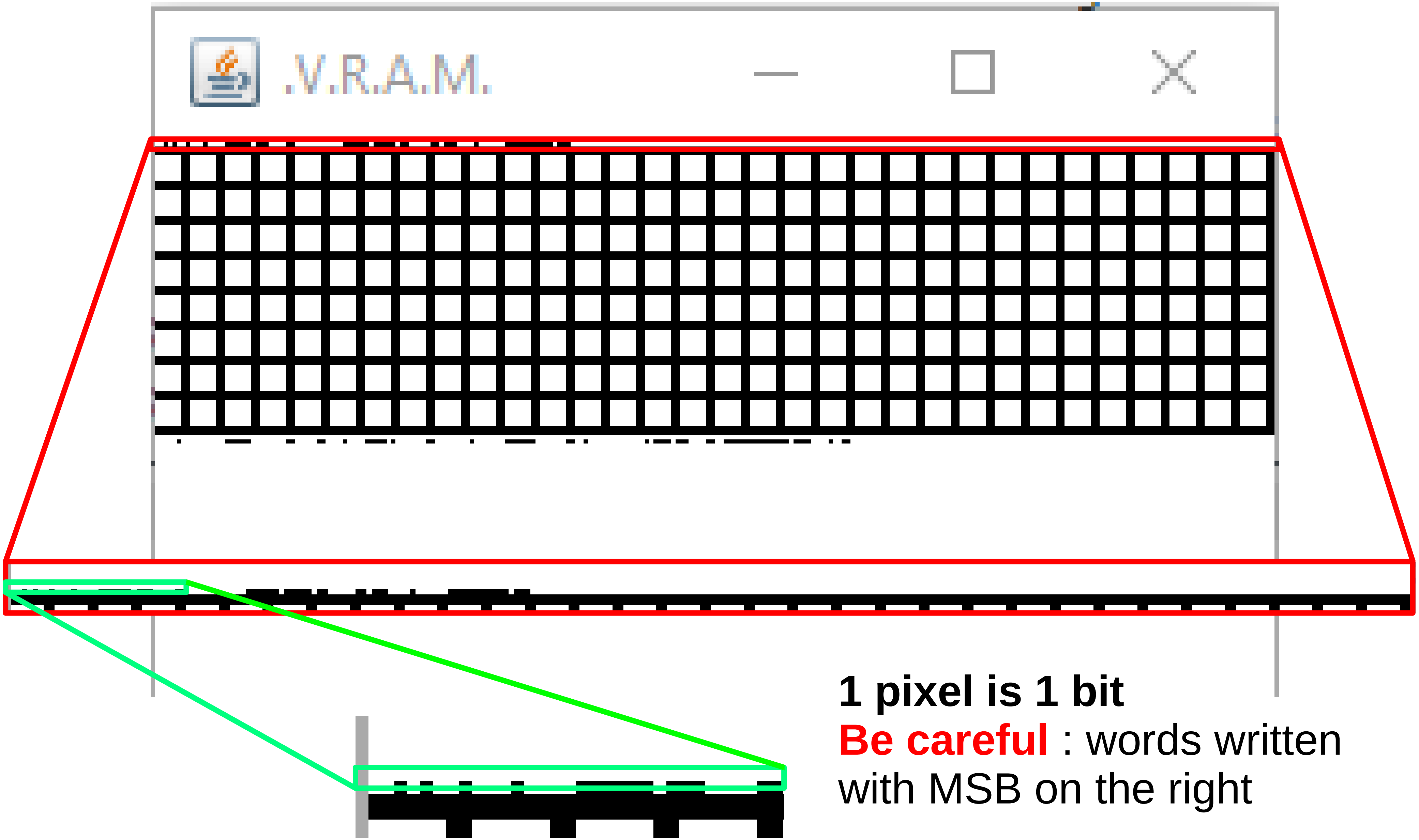
It displays the 1024 first words of the dynamic memory !

Memory view



One pixel line is composed of **8** 32-bits words.

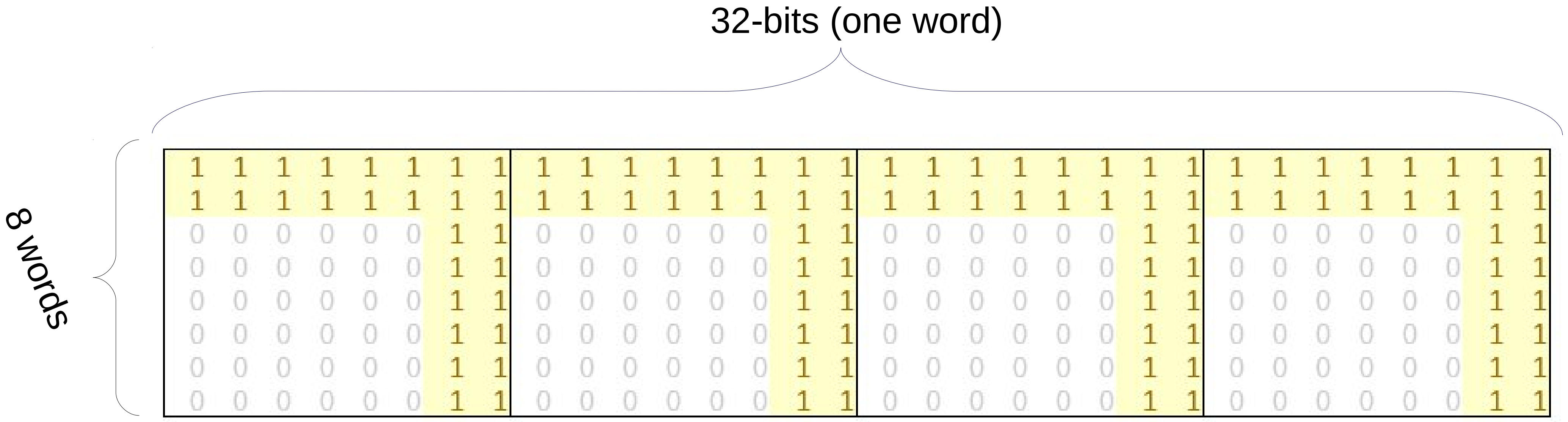
Memory view



1 pixel is 1 bit
Be careful : words written with MSB on the right

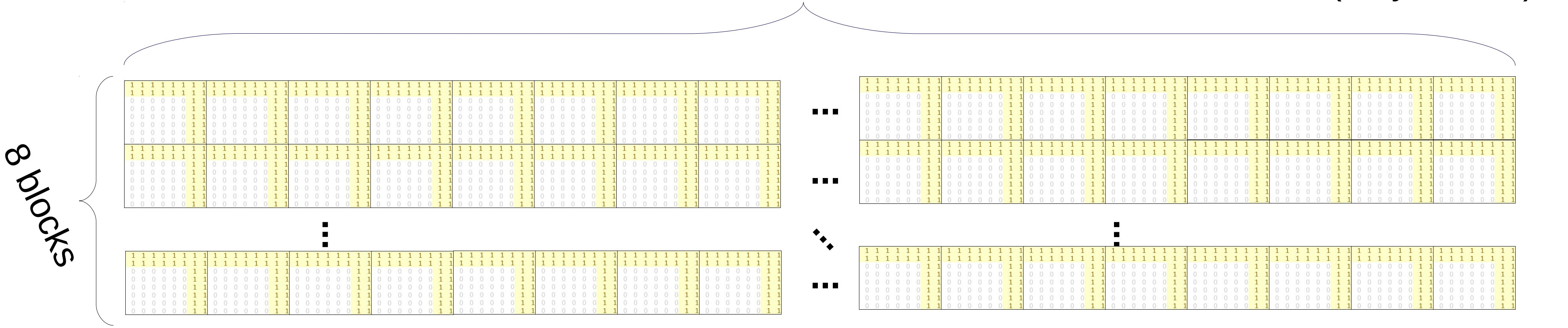
The maze in memory

(as displayed in the memory view of the simulator)



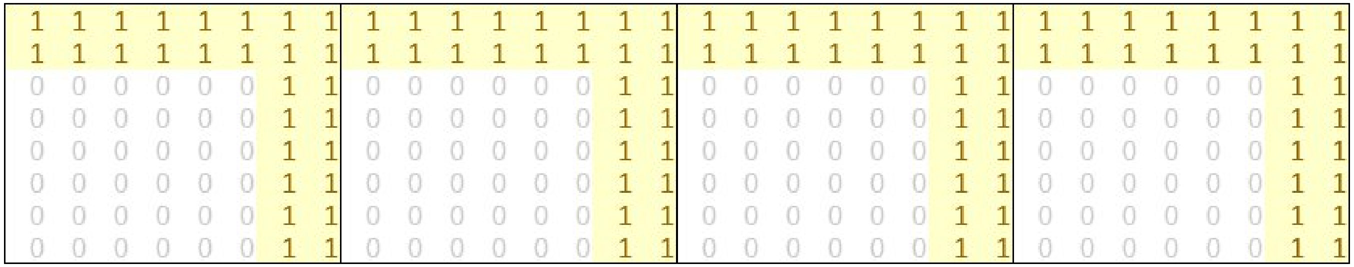
```
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
```

For this project, the maze will have
8 rows and 32 columns (i.e. 256 cells).



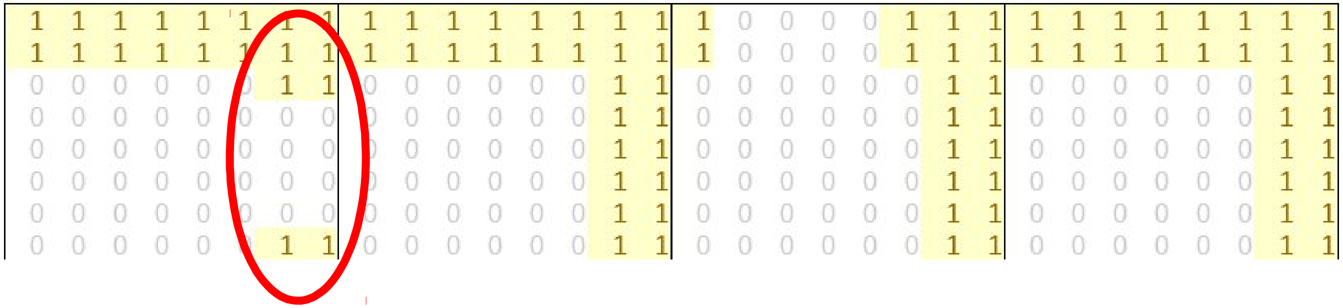
The maze in memory

(as displayed in the memory view of the simulator)



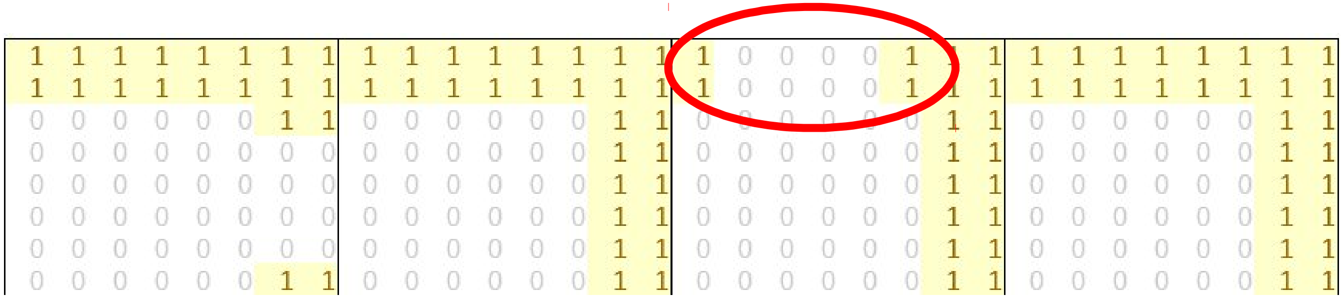
```
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
```

Open horizontal connection



```
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 0 0 0 0 1 1 1 | 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 0 0 0 0 1 1 1 | 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
```

Open vertical connection

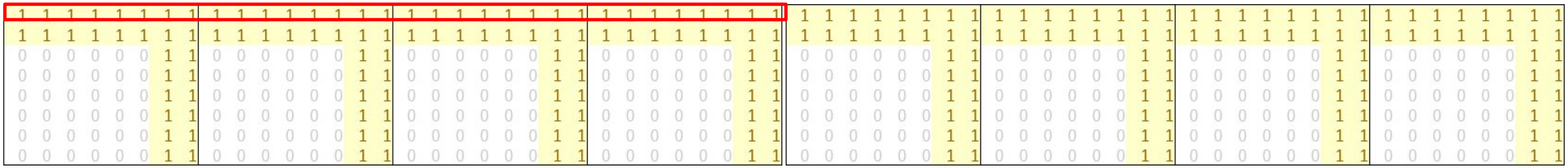


```
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 0 0 0 0 1 1 1 | 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 0 0 0 0 1 1 1 | 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
```

The maze in memory

(as displayed in the memory view of the simulator)

word 0, memory address : 64



```
1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  ...
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
```

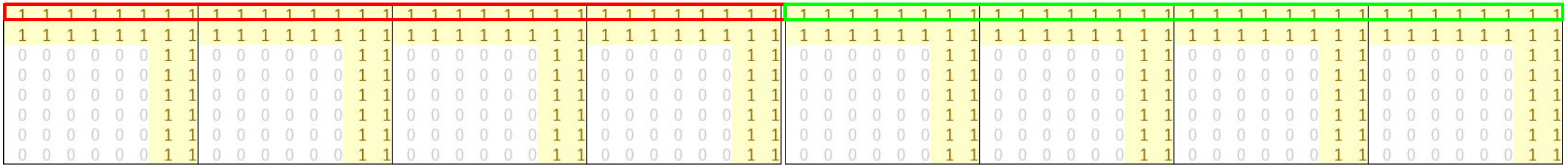
⋮ ⋮

The maze in memory

(as displayed in the memory view of the simulator)

word 0, memory address : 64

word 1, memory address : 68



```
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
```

...

⋮

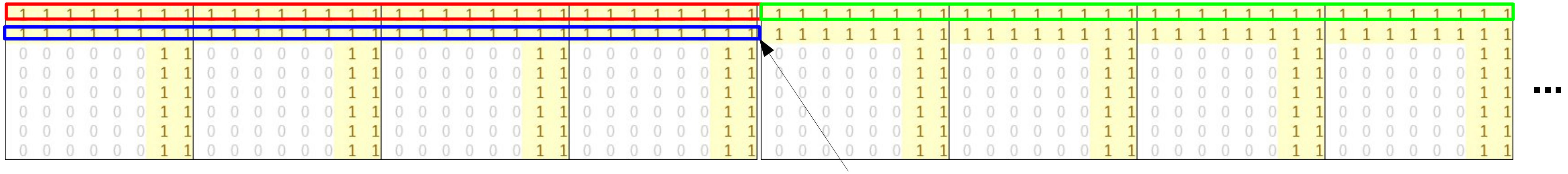
⋮

The maze in memory

(as displayed in the memory view of the simulator)

word 0, memory address : 64

word 1, memory address : 68



```
1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1  0 0 0 0 0 0 1 1
```

...

word 8, memory address : 96

⋮

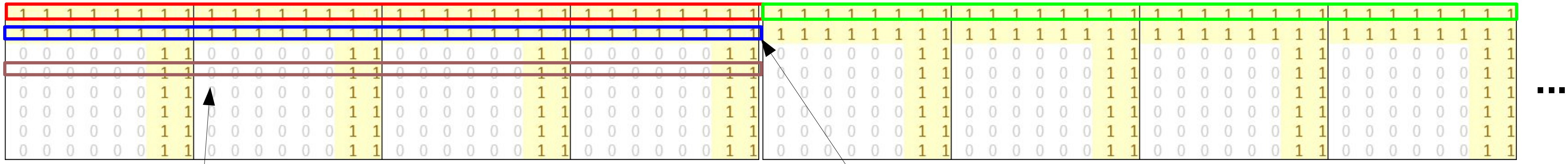
⋮

The maze in memory

(as displayed in the memory view of the simulator)

word 0, memory address : 64

word 1, memory address : 68



```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
```

...

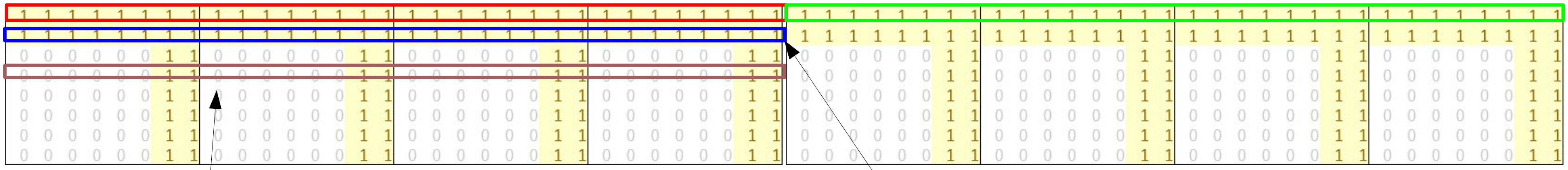
word 8, memory address : 96

word 24, memory address : 160

⋮ ⋮

The maze in memory

(as displayed in the memory view of the simulator)



```
1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1|1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1
```

word 0, memory address : 64

word 1, memory address : 68

...

word 8, memory address : 96

word 24, memory address : 160

⋮

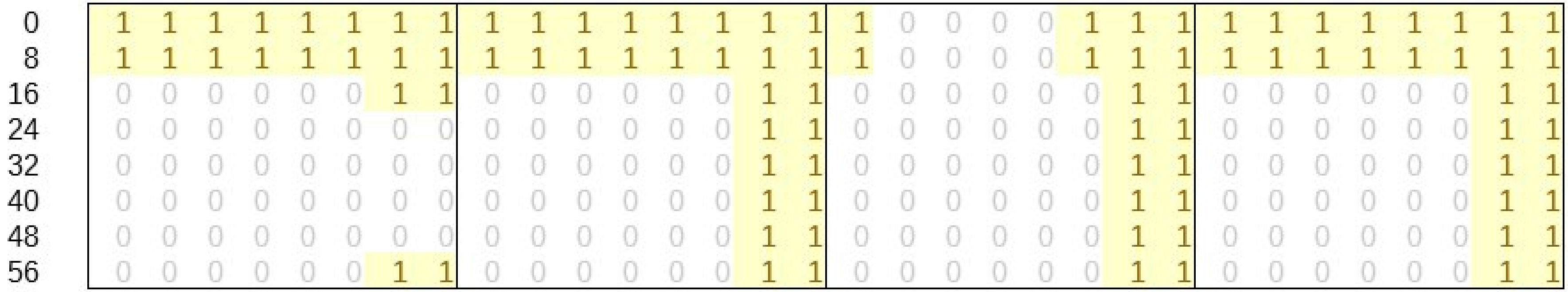
⋮

Hexadecimal representation :

- word 0 : `0xFFFFFFFF`
- word 24 : `0xC0C0C0C0`

The maze in memory

(as displayed in the memory view of the simulator)



```
 0   1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 0 0 0 0 1 1 1 | 1 1 1 1 1 1 1 1
 8   1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 0 0 0 0 1 1 1 | 1 1 1 1 1 1 1 1
16   0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
24   0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
32   0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
40   0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
48   0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
56   0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 1
```

Opening a door = applying a binary AND between some words and a mask

Horizontal conn. opening

```
word24 = word24 & 0xFFFFFF00
word32 = word32 & 0xFFFFFF00
word40 = word40 & 0xFFFFFF00
word48 = word48 & 0xFFFFFF00
```

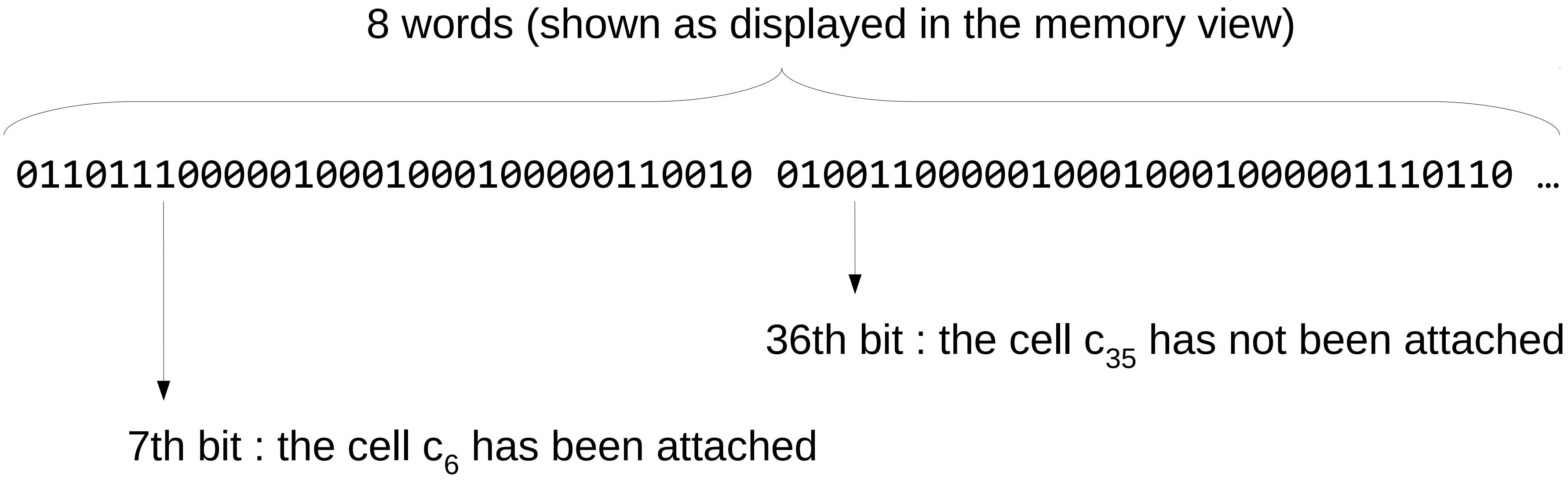
Vertical conn. opening

```
word0 = word0 & 0xFFE1FFFF
word8 = word8 & 0xFFE1FFFF
```

Be sure to apply the masking operations on the right byte

The `visited` bitmap

- The bitmap stores the information about cells that were already connected to the maze being built
- It consists of 8 words



8 words (shown as displayed in the memory view)

`01101110000010001000100000110010 01001100000100010001000001110110 …`

36th bit : the cell c_{35} has not been attached

7th bit : the cell c_6 has been attached

Coding guidelines

- Focus on code clarity and understandability before efficiency
- Still, your code shouldn’t be unreasonably inefficient (tip: use as few registers as possible, avoid repeating useless operations)
- Document your code !!
- Procedures and macros should be documented:
  - Parameters
  - Operations performed

Files and submission

You are provided with:

- **perfect_maze.c** : a C implementation of the maze construction algorithm. You can use it as basis for your assembly implementation.
- **beta.usam** : definition of the beta-assembly. Check this file to see which macro you can use.
- **main.asm** : this file contains the main program that will be used to test your procedure

You must submit **in a ZIP file named « sXXXXXX_NAME1_sYYYYYY_NAME2.zip »**:

- **perfect_maze.asm** : a file containing your implementation of the maze construction algorithm
- **(optional) report.pdf** : if you think you need more than the comments to explain some parts of your code, you can write those explanations in a short report (maximum two pages).
- **Submitting other files will be sanctioned**