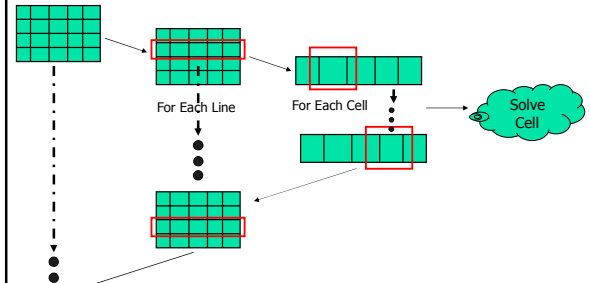


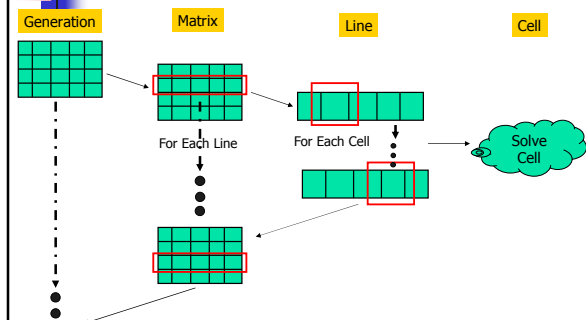
Toward a general purpose computer

Example: Game of Life

Game of Life



Game of Life



Modularity

- Generation Module
- Matrix Module
- Line Module
- Cell Module

Algorithm for the Generation Module

1. CurrentGen = 0
2. **Solve for current Matrix**
3. CurrentGen++
4. If CurrentGen < MAX_GEN
goto 2

Solve Matrix

0. j = 1
1. Calculate line (j)
2. If j < MAX_LINE
 - 2.1 j++
 - 2.2 goto 1.
3. STOP

Calculate Line j

0. $i = 0$, $tmp = 0$
1. Calculate Cell i .
2. If alive, set location i in tmp to alive
3. If $i < MAX_CELL$
 - 3.1 $i++$
 - 3.2 goto 1.
4. Store tmp in :
 - even generation: $i+16$
 - odd generation : $i-16$

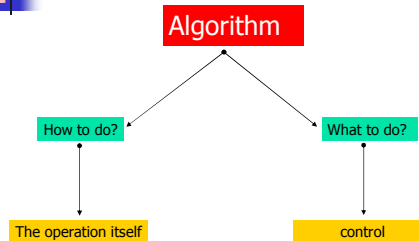
CellAlive(cell index i , line index j)

0. $N = \text{Count_Neighbors}(i,j-1,\text{false})$
1. $N = N + \text{Count_Neighbors}(i,j,\text{true})$
2. $N = N + \text{Count_Neighbors}(i,j+1,\text{false})$
3. If $N > MIN \ \&\& \ N < MAX$
 - 3.1 Return alive
4. Return dead

Count_Neighbors (index i ,line j , center)

0. $N = 0$
1. If alive in location $[i-1,j]$: $N=N+1$
2. If not center:
 - 2.1 If alive in location $[i,j]$: $N=N+1$
3. If alive in location $[i+1,j]$: $N=N+1$
4. Return N

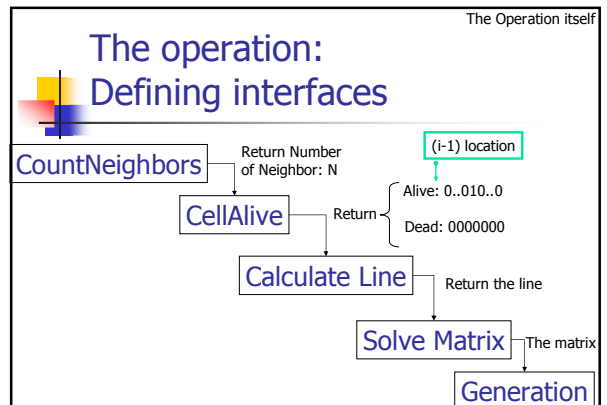
Algorithm implementation



The operations in the algorithm:

- Operations:
- Add
 - Subtract

The operation: Defining interfaces



The Operation itself

The operation:

Module Calculate Line, instruction 2

If alive, set location i in tmp to alive

If alive

Calculate_Line(j) = 0001000
OR
tmp = 00000000

Result = 000100000

If dead

Calculate_Line(j) = 0000000
OR
tmp = 00000000

Result = 000000000

The j-th location

The Operation itself

The operation:

Module Calculate Line, instruction 2

If alive, set location i in tmp to alive

↓

tmp = OR(tmp, Calculate_Line(j))

The Operation itself

The operations in the algorithm:

Operations:

- Add
- Subtract
- OR

The Operation itself

Implementing instructions

Module Count_Neighbors, instruction 1, 2.1, 3

If alive in location [i-1,j]

decode(i-1) = 0001000
AND
Line (j) = 00010100

Result = 00010000

Non Zero – cell in location i is alive

decode(i-1) = 0001000
AND
Line (j) = 00000100

Result = 00000000

Zero – cell in location i is dead

The Operation itself

Implementing instructions

Module Count_Neighbors, instruction 1, 2.1, 3

If alive in location [i-1,j]

↓

Alive if AND(line, decode(i-1)) is not zero

The Operation itself

The operations in the algorithm:

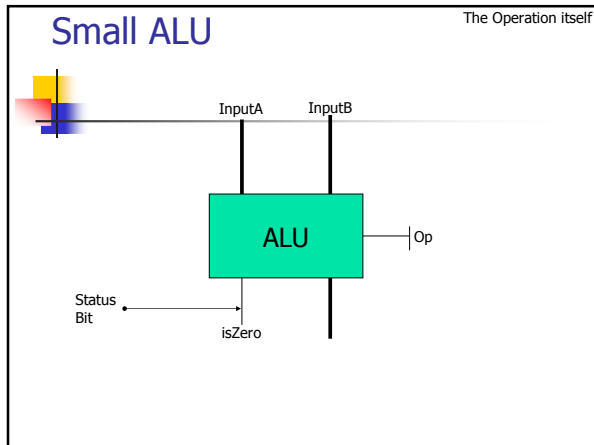
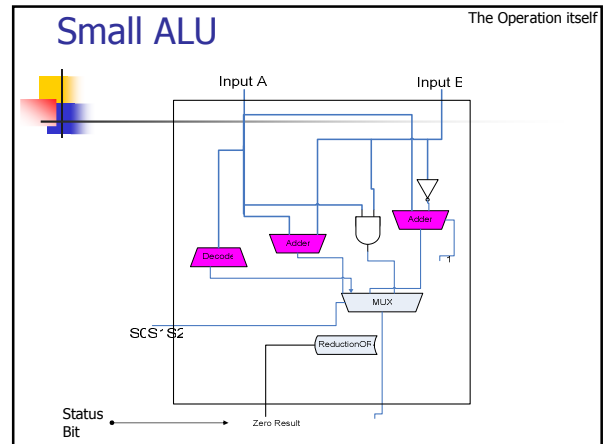
Operations:

- Add
- Subtract
- The logic AND + Check if zero

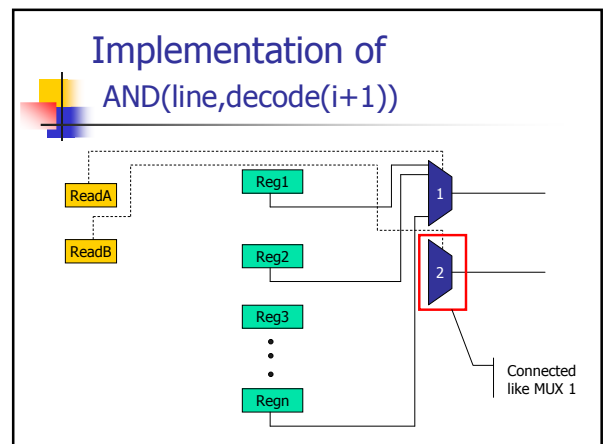
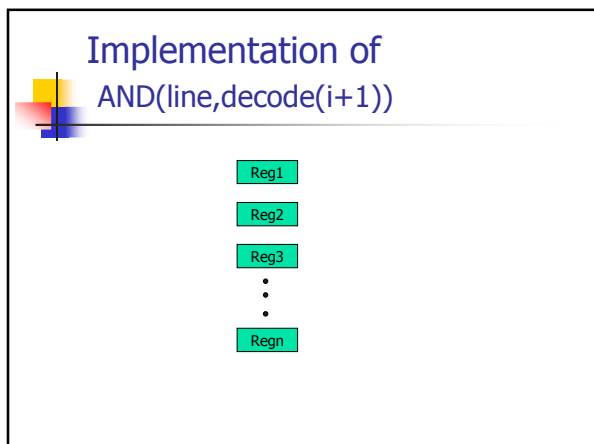
Small ALU (arithmetic logic unit)

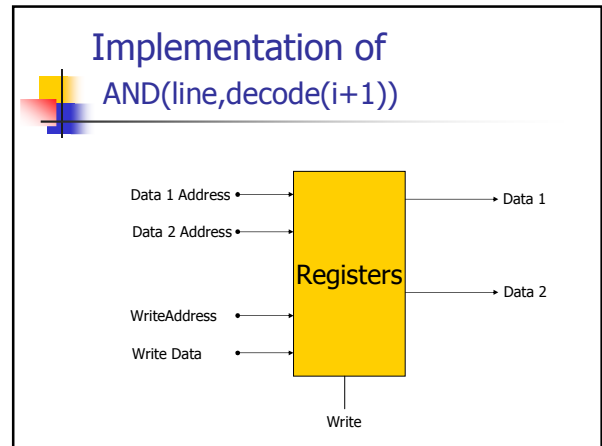
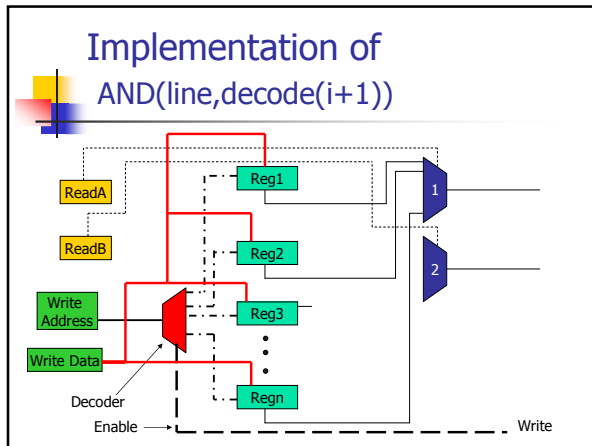
The Operation itself

Op selector	Meaning	Result
00	Add	Res = InputA+InputB
01	Subtract	Res = InputA+ NOT(InputB)+1
10	AND	Res = AND(InputA,InputB)
11	Decode	Res = decode(InputA)



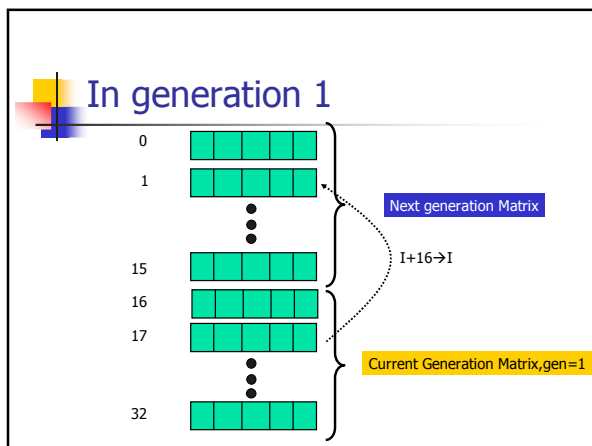
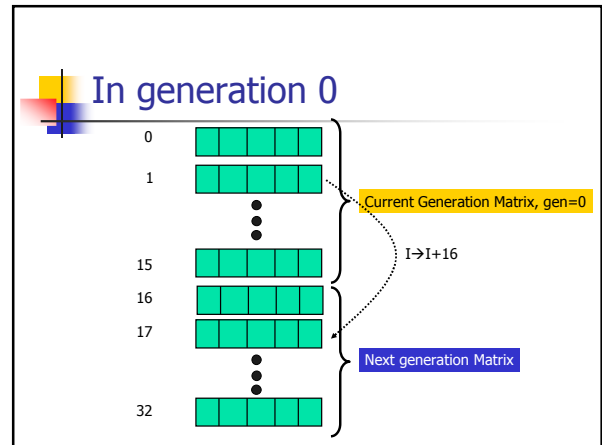
- ### Variables of the algorithm
- #### Count_Neighbors
- Line j – The current line
 - i - The index we are processing
 - N - The number of Neighbors so far
 - tmp - temporary processing space





Variables of the algorithm Count_Neighbors

Variable	Meaning	Address
CurrentLine	Hold the current line	0
I	The current index	1
N	The Neighbor	2
Tmp	Temporary space	3
Constant 1	Constant 1	5
ControlVariable	A flag to determine if the operation should be performed	6
Center	Is it the center	7



Variables of the algorithm Count_Neighbors

Variable	Meaning	Address
CurrentLine	Hold the current line	0
I	The current index	1
N	The Neighbor	2
Tmp	Temporary space	3
Constant 1	Constant 1	5
ControlVariable	A flag to determine if the operation should be performed	6
Center	Is it the center	7

