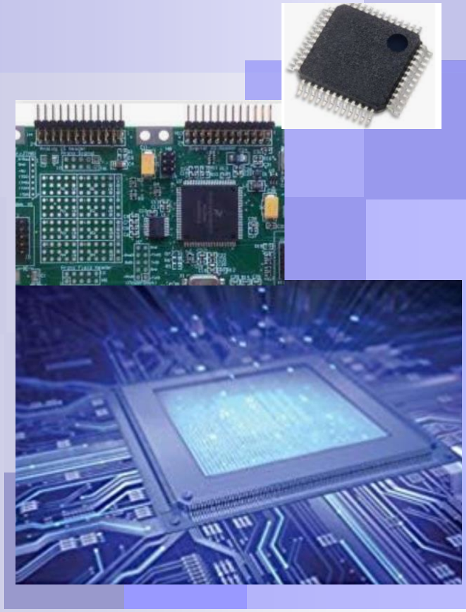
CE320 : Microcomputers I



# Lab3
# Assembly Loop Programming and Array

**Jung Me Park, Ph.D**

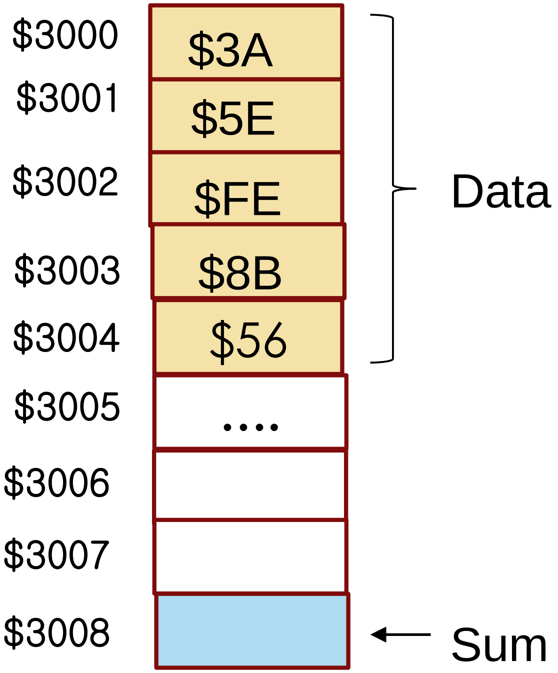
**ECE Department, Kettering University**

**jpark@kettering.edu**

Kettering University | College of Engineering

# *Part 0. A looping program to add five numbers.*

- Write a looping program that adds a list of five one-byte numbers **beginning at address $3000**, and stores the one-byte **result (sum) at address $3008**.

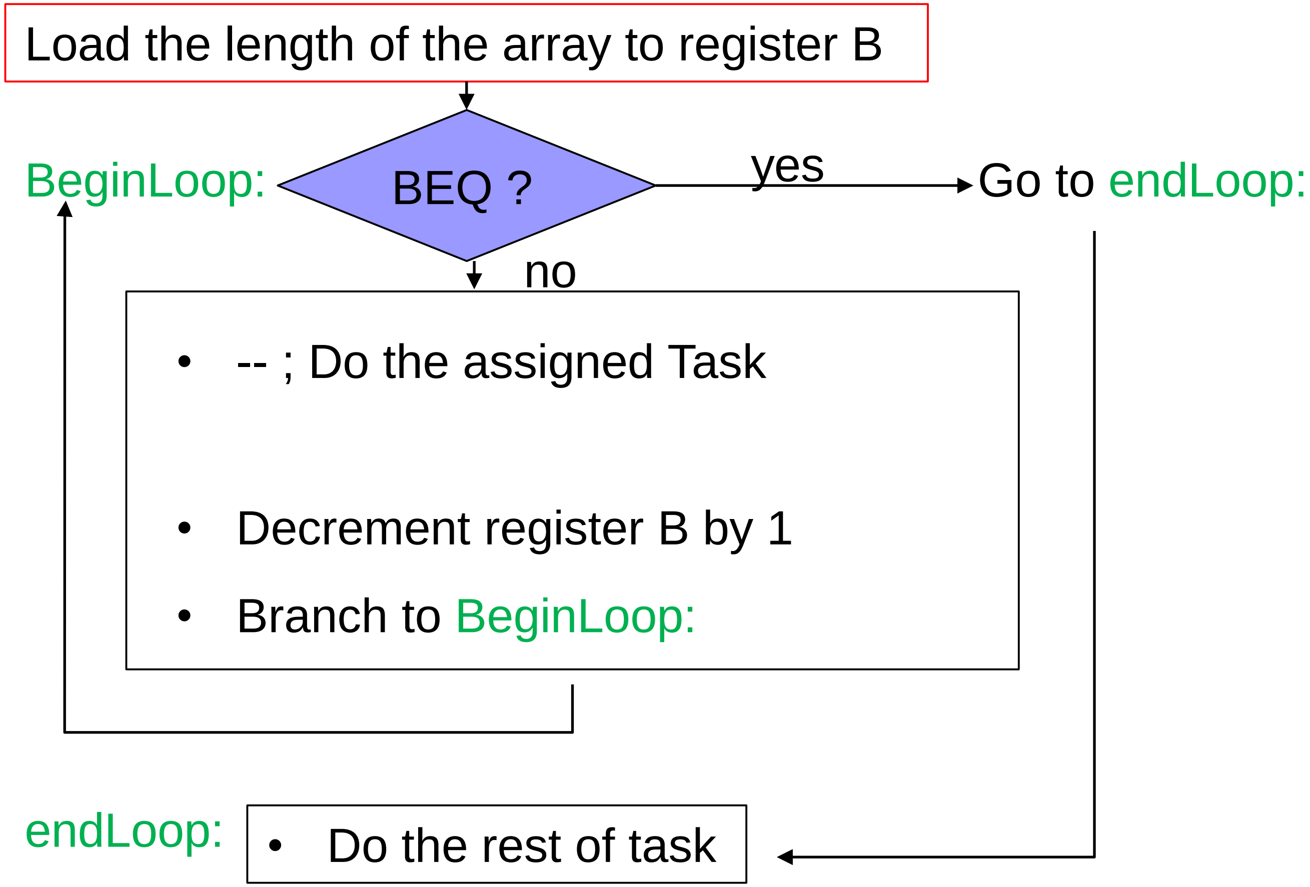


|  |  |
|---|---|
| $3000 | $3A |
| $3001 | $5E |
| $3002 | $FE |
| $3003 | $8B |
| $3004 | $56 |
| $3005 | …. |
| $3006 |  |
| $3007 |  |
| $3008 |  |

  - ○ a list of five one-byte numbers starting **at address $3000**
  - ○ stores the one-byte **result in address $3008**

```
         ORG      $3000
Array    DC.B  $3A, $5E, $FE, $8B, $56
```

# Loop Structure in Assembly (I)



Load the length of the array to register B

BeginLoop:

BEQ ?

yes

Go to endLoop:

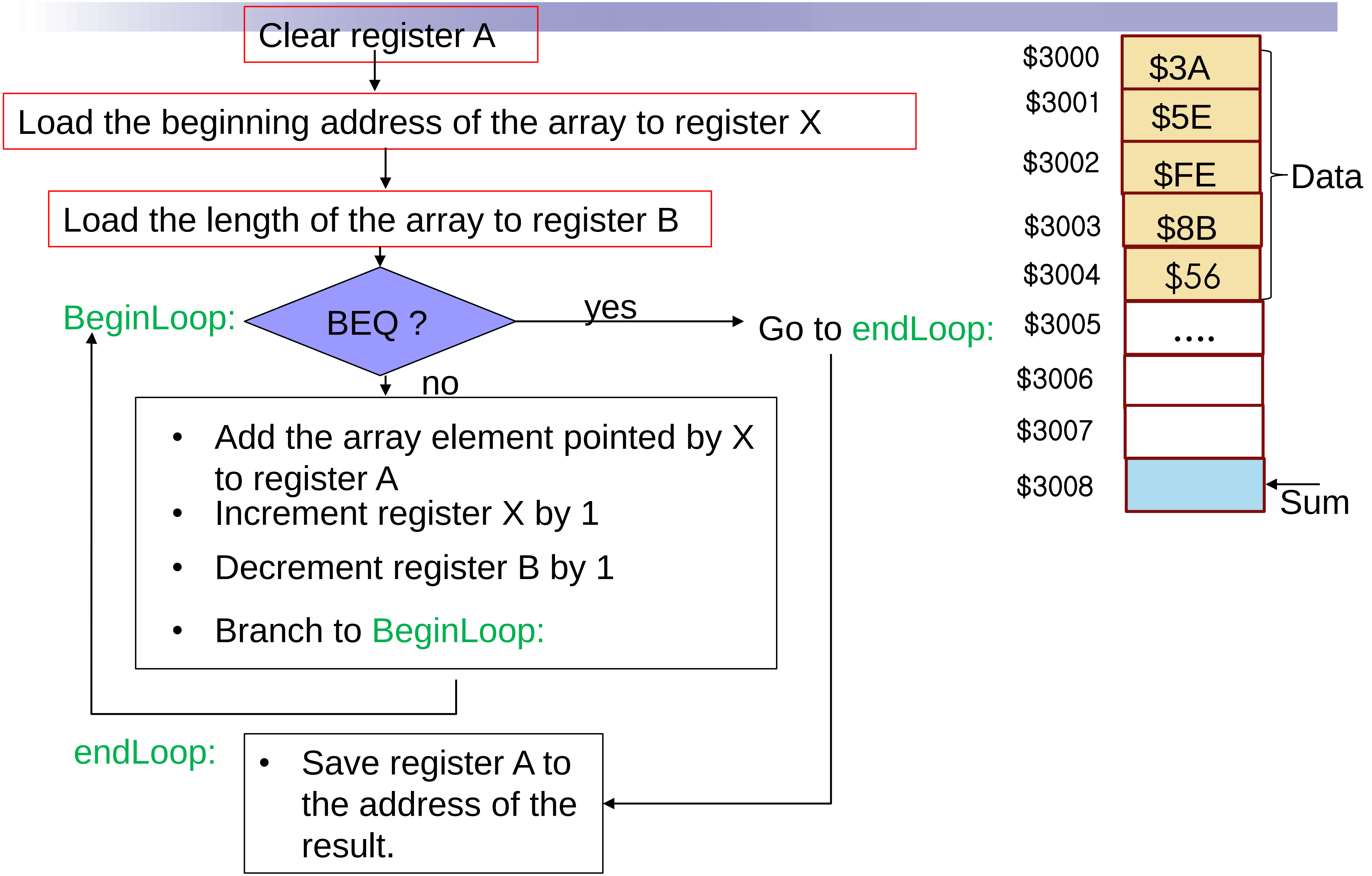
no

- -- ; Do the assigned Task
- Decrement register B by 1
- Branch to BeginLoop:

endLoop:

- Do the rest of task

# Loop Structure in Assembly (II)



Clear register A

Load the beginning address of the array to register X

Load the length of the array to register B

BeginLoop: BEQ ?

yes → Go to endLoop:

no

- Add the array element pointed by X to register A
- Increment register X by 1
- Decrement register B by 1
- Branch to BeginLoop:

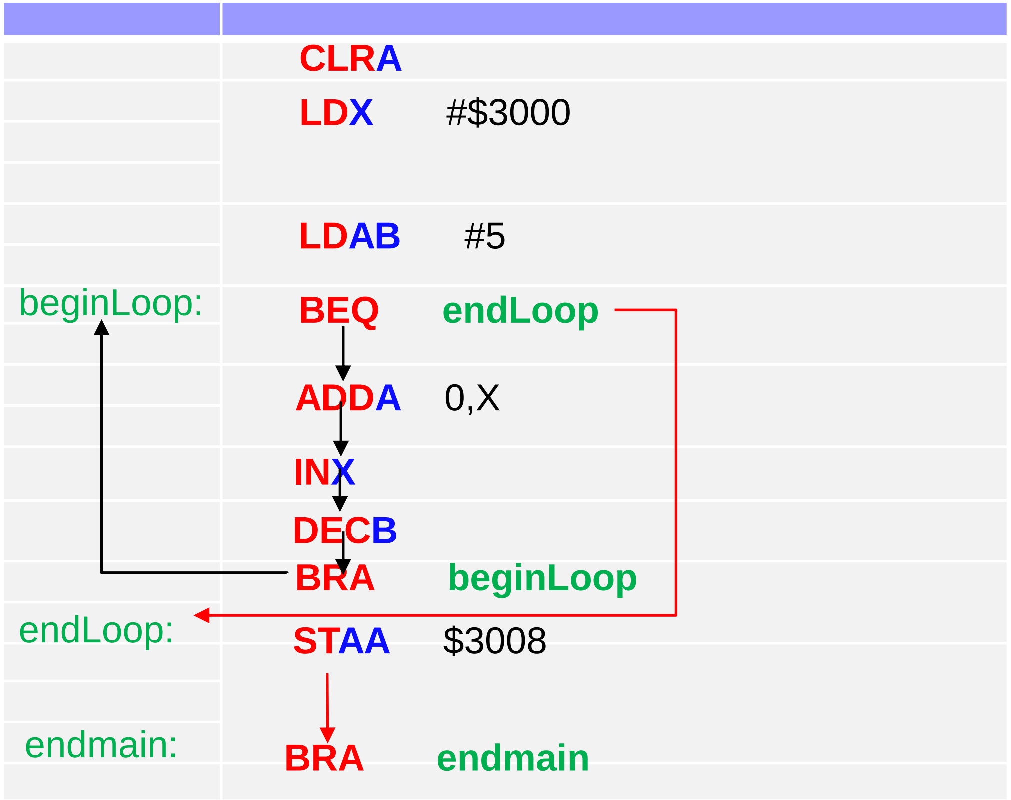
endLoop:

- Save register A to the address of the result.

| | | |
|---|---|---|
| $3000 | $3A | Data |
| $3001 | $5E | Data |
| $3002 | $FE | Data |
| $3003 | $8B | Data |
| $3004 | $56 | Data |
| $3005 | …. | |
| $3006 | | |
| $3007 | | |
| $3008 | | Sum |

# Part 0. A looping program to add five numbers.

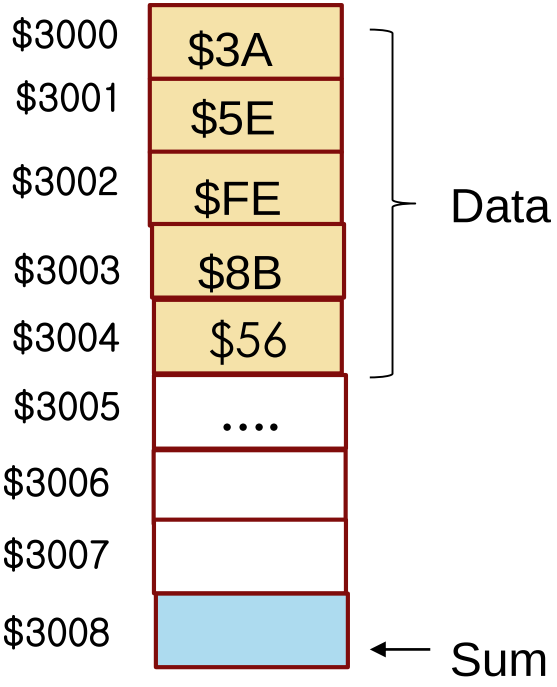
- To calculate the sum of the array elements using the looping structure, the following registers are used:
  - Index register, **X** : hold the address of the array element
  - Accumulator register, **B** : count the number of iterations in the loop
  - Accumulator register, **A** : add the array elements.



```
            CLRA
            LDX     #$3000

            LDAB    #5
beginLoop:  BEQ     endLoop
            ADDA    0,X
            INX
            DECB
            BRA     beginLoop
endLoop:    STAA    $3008

endmain:    BRA     endmain
```



| | |
|---|---|
| $3000 | $3A |
| $3001 | $5E |
| $3002 | $FE |
| $3003 | $8B |
| $3004 | $56 |
| $3005 | .... |
| $3006 | |
| $3007 | |
| $3008 | |

# Part 0. A looping program to add five numbers.

| | | |
|---|---|---|
| | CLRA | A=$00 |
| | LDX #$3000 | X=$3000 |
| | LDAB #5 | B=$05 ; change N=0 Z=0 V=0 |
| beginLoop: | BEQ endLoop | BEQ is not taken because z=0 |
| | ADDA 0,X | A= A + mem[0+X] =$00+$3A |
| | INX | X=X+1= $3001 |
| | DECB | B=B-1 = $04 ; change N=0 Z=0 V=0 C=0 |
| | BRA beginLoop | go to beginLoop: |
| endLoop: | STAA $3008 | |
| endmain: | BRA endmain | |

| | |
|---|---|
| $3000 | $3A |
| $3001 | $5E |
| $3002 | $FE |
| $3003 | $8B |
| $3004 | $56 |
| $3005 | .... |
| $3006 | |
| $3007 | |
| $3008 | |

# Part 0. A looping program to add five numbers.

In the previous iteration
[A]=$3A
[B]=$04
[X]=$3001
N=0  Z=0 V=0 C=0

| | | |
|---|---|---|
| | CLRA | |
| | LDX  #$3000 | |
| | LDAB #5 | |
| beginLoop: | BEQ  endLoop | BEQ is not taken because z=0 |
| | ADDA  0,X | A= A + mem[0+X] =$3A+$5E |
| | INX | X=X+1= $3002 |
| | DECB | B=B-1 = $03  ; change N=0  Z=0 V=0 C=0 |
| | BRA beginLoop | go to beginLoop |
| endLoop: | STAA $3008 | |
| endmain: | BRA endmain | |

| | |
|---|---|
| $3000 | $3A |
| $3001 | $5E |
| $3002 | $FE |
| $3003 | $8B |
| $3004 | $56 |
| $3005 | .... |
| $3006 | |
| $3007 | |
| $3008 | |

# Part I: Working with Array

- **Array:**
  - a group of consecutive memory locations.
  - Note that arrays are created in the *data* section.
  - As part of array specification, we need to know **the length of array**, or in case of a text array, a NULL character may be used as the end of array.

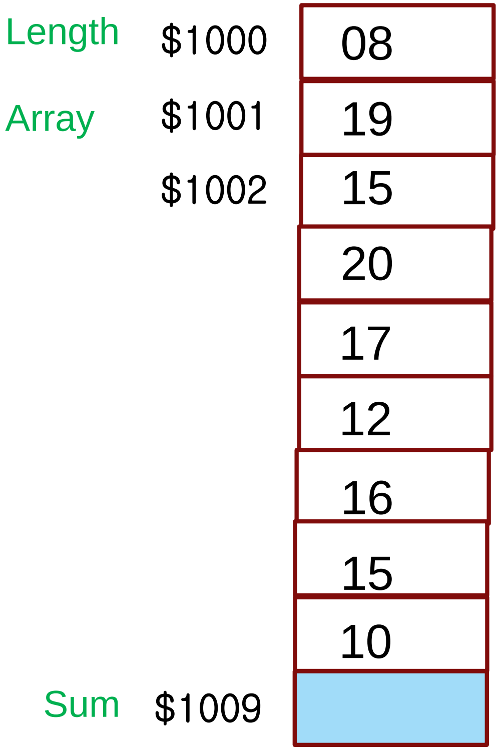
- **Array processing:** **Indexed addressing mode** is a must-have feature of underlying microprocessor to access array elements inside the loop.

- *Write an assembly program that computes* ***the sum of the given array using loop structure.***

```
;***********Enter your data here:
                ORG   $1000
Length:         DC.B    8                                   ; the length of the array
Array:          DC.B    19, 15, 20, 17, 12, 16, 15, 10      ; elements in the array
Sum:            DS.B    1                                   ; the sum is stored here
```

# Part I: Working with Array

```
;***********Enter your data here:
                ORG    $1000
Length:         DC.B     8                                  ; the length of the array
Array:          DC.B     19, 15, 20, 17, 12, 16, 15, 10     ; elements in the array
Sum:            DS.B     1                                  ; the sum is stored here
```



| | | |
|---|---|---|
| Length | $1000 | 08 |
| Array | $1001 | 19 |
| | $1002 | 15 |
| | | 20 |
| | | 17 |
| | | 12 |
| | | 16 |
| | | 15 |
| | | 10 |
| Sum | $1009 | |

; Data is provided in decimal

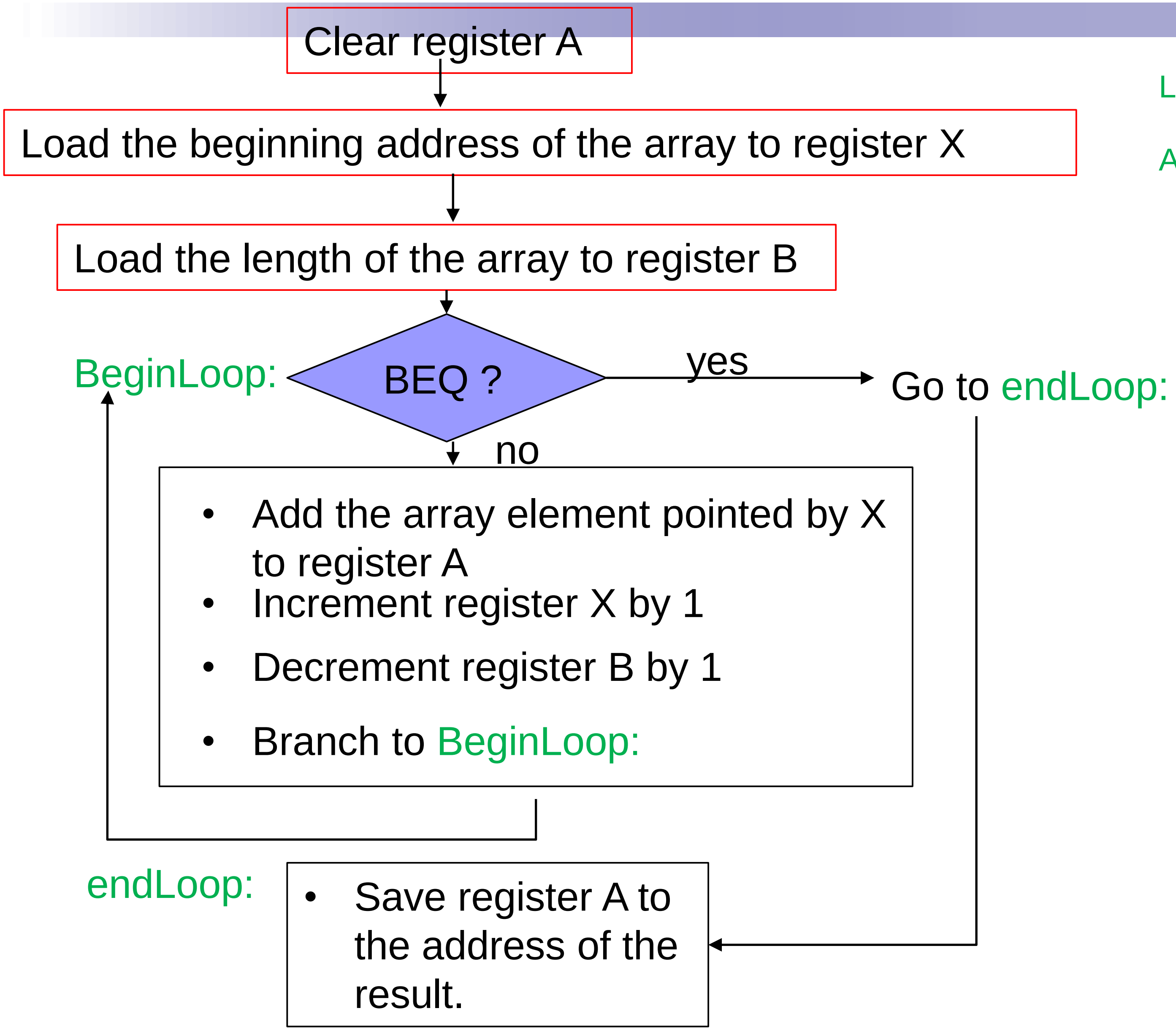
# Part I: Working with Array

- Labels: Meaningful name of the address
  - **Length** =$1000
  - **Array** = $1001
  - **Sum** = $1009

| | | |
|---|---|---|
| Length | $1000 | 08 |
| Array | $1001 | 19 |
| | $1002 | 15 |
| | | 20 |
| | | 17 |
| | | 12 |
| | | 16 |
| | | 15 |
| | | 10 |
| Sum | $1009 | |

- Use labels in your program
  - **LDAB** $1000 → **LDAB Length**
  - **LDX** #$1001 → **LDX #Array**
  - **STAA** $1009 → **STAA Sum**

# Part I: Working with Array



Clear register A

Load the beginning address of the array to register X

Load the length of the array to register B

BeginLoop:

BEQ ?

yes

Go to endLoop:

no

- Add the array element pointed by X to register A
- Increment register X by 1
- Decrement register B by 1
- Branch to BeginLoop:

endLoop:

- Save register A to the address of the result.

| | | |
|---|---|---|
| Length | $1000 | 08 |
| Array | $1001 | 19 |
| | $1002 | 15 |
| | | 20 |
| | | 17 |
| | | 12 |
| | | 16 |
| | | 15 |
| | | 10 |
| Sum | $1009 | |

# Part II:

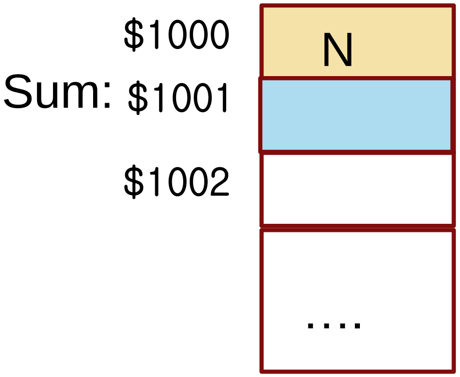
- ***Write a program that computes the sum of the first N natural numbers (i.e., Sum = 1 + 2 + 3 + … + N)****.*

- *Assume N is a one byte unsigned integer number whose value is given at memory location $1000.*
  - *N <= 20 , so you could safely store (without overflow) the result of the Sum in a one byte memory location at address $1001**.***

Tips:

-Use register B as a counter
: use Decrement/or increment
- Use register A to calculate the sum
: Addition Instruction
- Use branch instructions for the loop

- Save the sum in the memory, $1001



$1000 N

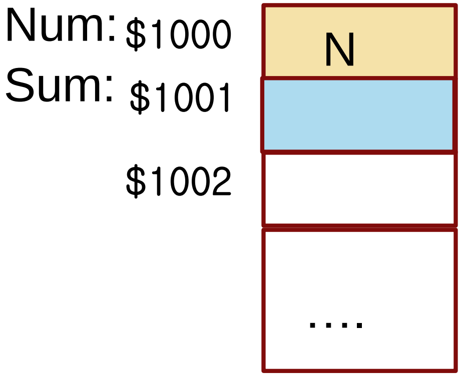
Sum: $1001

$1002

….

# Part II:

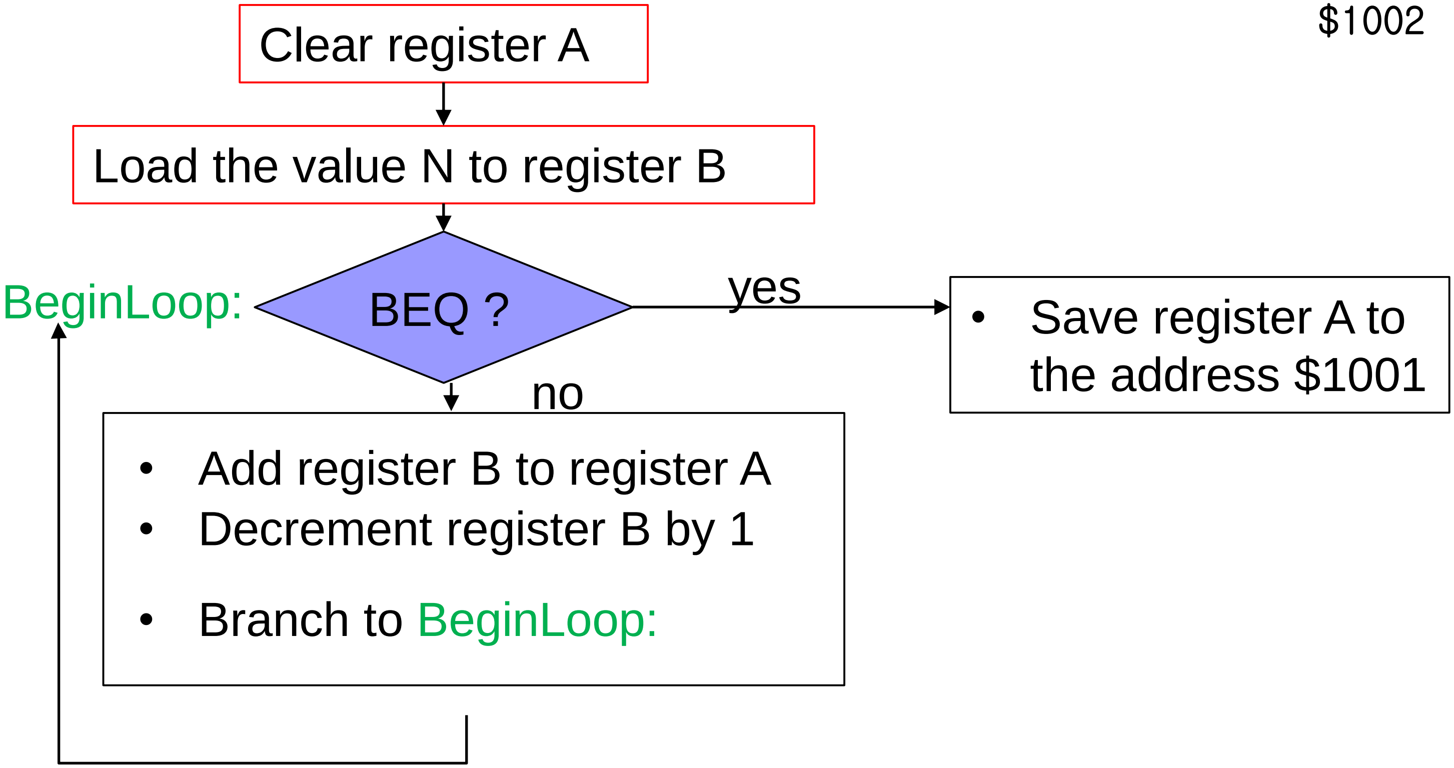
- ***Write a program that computes the sum of the first N natural numbers (i.e., Sum = 1 + 2 + 3 + … + N).***
- *Program Flow Chart*


Num: $1000 N

Sum: $1001

$1002

….


Clear register A

Load the value N to register B

BeginLoop:

BEQ ?

yes

- Save register A to the address $1001

no

- Add register B to register A
- Decrement register B by 1
- Branch to BeginLoop: