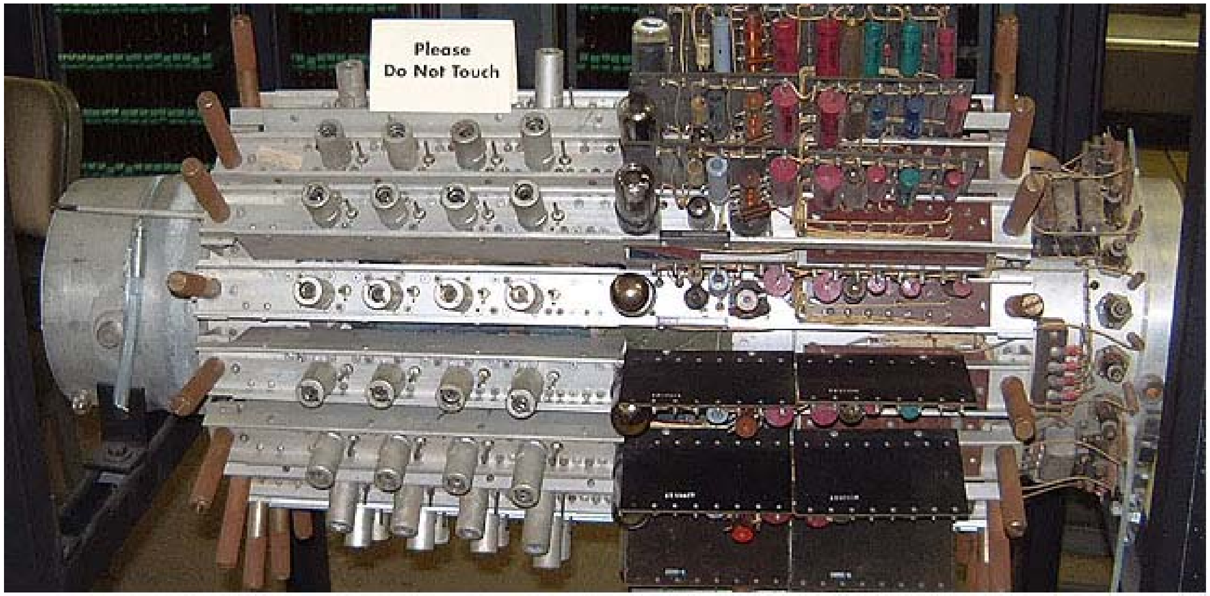
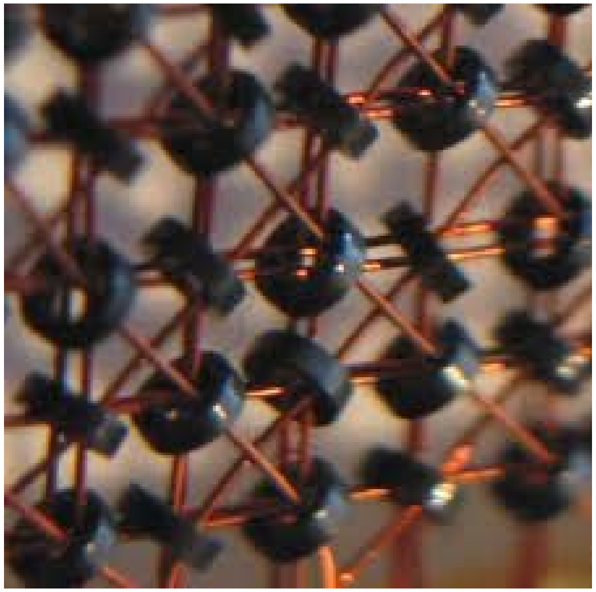
04 Memory Organization

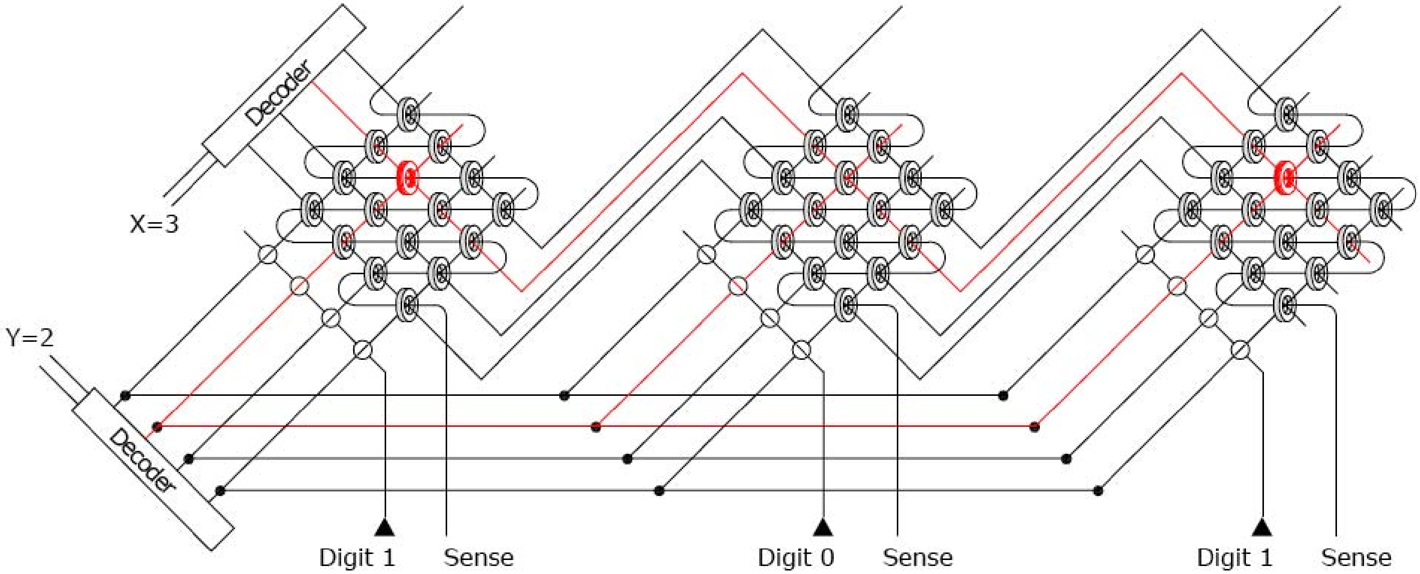
CSC 230



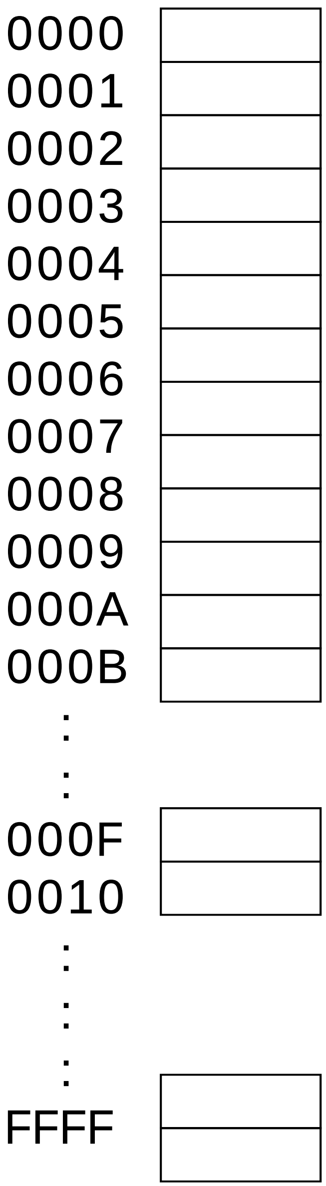
Do you know what this is?



Core Memory


Decoder
X=3
Y=2
Decoder
Digit 1
Sense
Digit 0
Sense
Digit 1
Sense

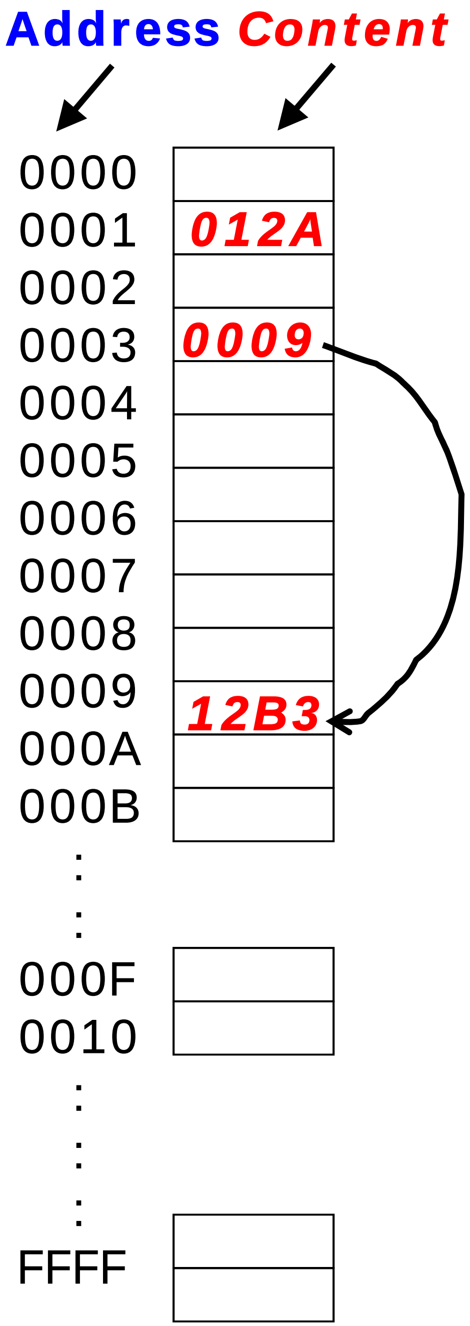
A Conceptual Model of Memory Space



Memory is an array

- **each location has a unique address**
- **the address is the index that identifies the location in the memory space**
- **each location holds a fixed amount of information**
- **a 16-bit address allows for 65,636 (64K) addressable locations**
  ➔ *WHY?*

What is it Meant by "Memory Addressing"?



- **memory is a sequence of directly addressable "locations"**
- **content of each location is just bits**

  ➔ **they can be viewed as data, instructions, other addresses, etc.**
- **any piece of information in memory has 2 components:**

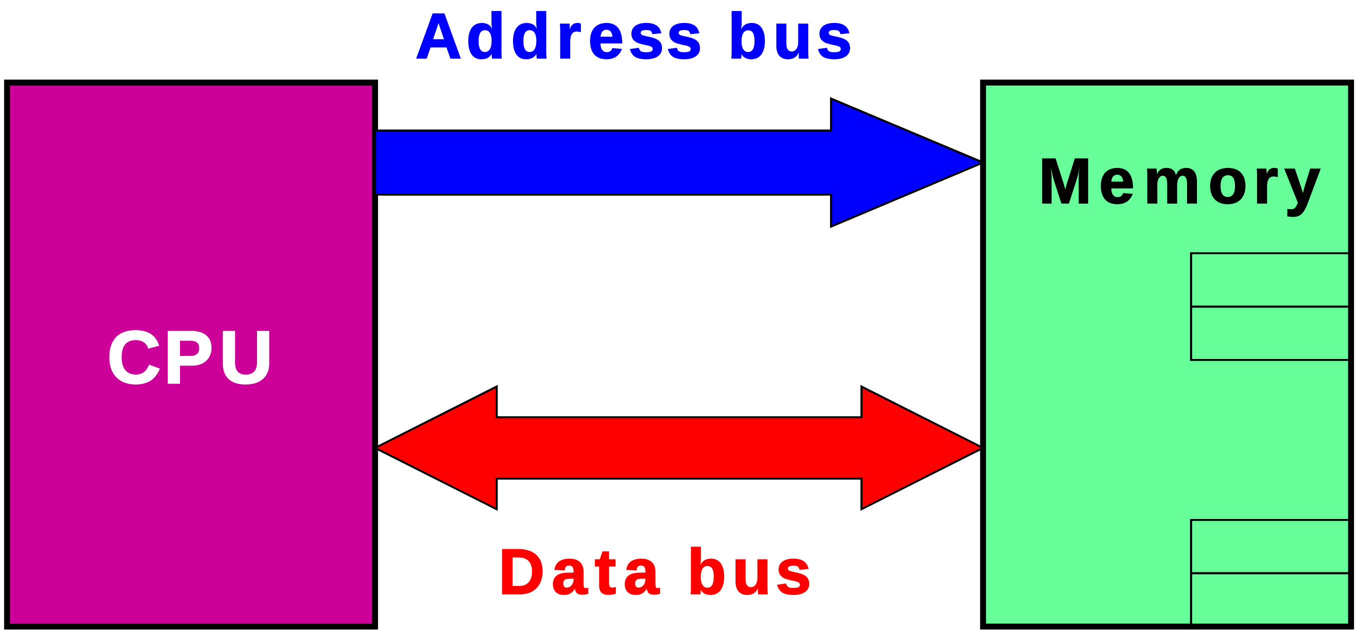
  ➔ **address and content**

Address → **Content**

the address is a pointer to the location where we want to use the content

Memory Addressing

- CPU supplies the address on the address bus
- The content for a location is transmitted on the data bus
- A transfer of data *from* memory is a **read**.
- A transfer of data *to* memory is a **write**.



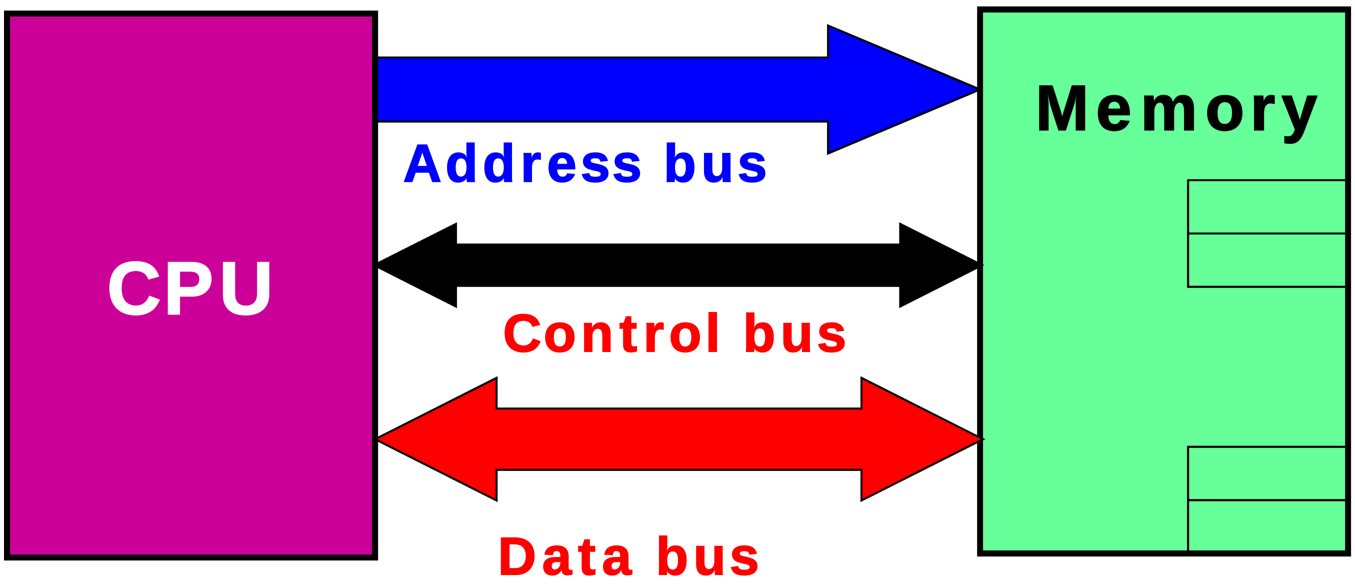
Memory Addressing, Read & Write: the Sequence

READ

1. CPU supplies the address on the address bus
2. READ signal is placed on the control bus
3. Wait for memory to respond
4. The content is transmitted on the data bus from memory to CPU

WRITE

1. CPU supplies the address on the address bus
2. CPU places content for the given location on the data bus
3. WRITE signal is placed on the control bus



Addressing – the overall view

1. **Memory organization implies:**
   - Sequence of locations of same size
   - Consecutive bytes form words, half words, double words.
   - There is a unique address for each location

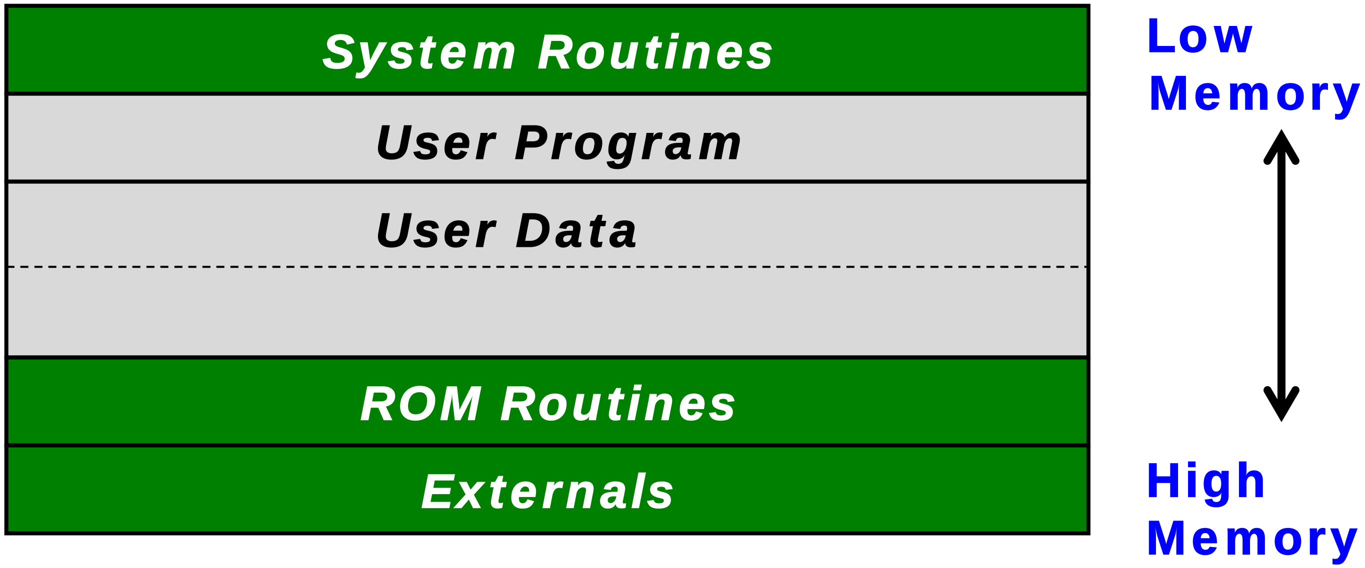
2. **Size of the address bus**
   - determines the *address space* of the system,
   - that is, the total number of addressable locations
   - It does NOT say anything about the size of each location

3. **n-bit address bus can point to 2^n distinct locations**

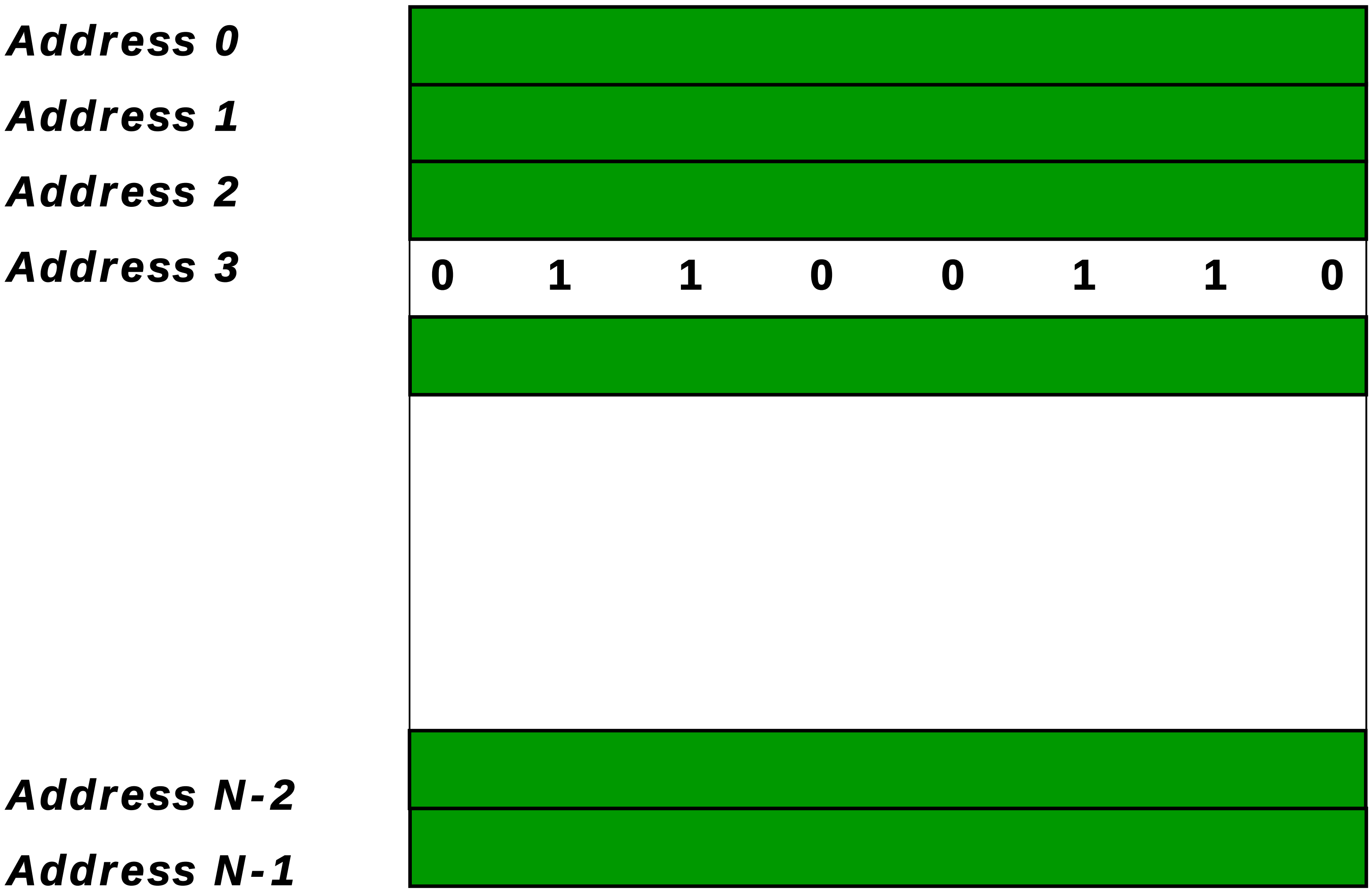
4. **the address space of the system (that is, the total number of unique addresses) is not the same as the amount of actual memory**
   - With an n-bit address bus we should not buy a memory with 2^n locations ➔ no room left in the addressable space for peripherals

5. **Size of the data bus**
   - May indicate the size of each location

Typical Memory Map



Addressing and Pointers



Addressing and Pointers: by example

Register	Value
R1	*0000C00C*
R2	00000000
R3	*0000C018*
R4	00000000
R5	00000000

memory addresses	memory content
0xC000	028F101C
0xC004	06112000
0xC008	028F3018
0xC00C	06134000
0xC010	00825004
0xC014	028F6010
0xC018	02165000
0xC01C	00000005
0xC020	00000006
0xC024	00000000

Addressing and Pointers : by example

```
LDR R2,[R1]   @ R2 ← mem[R1]
              @ that is, copy (load) into R2
              @ the content of memory
              @ at the location whose address
              @ is contained in R1

LDR R4,[R3]   @ R4 ← mem[R3]
              @ that is, copy (load) into R4
              @ the content of memory
              @ at the location whose address
              @ is contained in R3

ADD  R5,R2,R4  @ R5 = R2+R4
```

Addressing and Pointers: by example

CPU

R1	*0000C00C*
R2	00000000
R3	*0000C018*
R4	00000000
R5	00000000

RAM

memory addresses	memory content
0xC000	028F101C
0xC004	06112000
0xC008	028F3018
0xC00C	06134000
0xC010	00825004
0xC014	028F6010
0xC018	02165000
0xC01C	00000005
0xC020	00000006
0xC024	00000000

Addressing and Pointers: by example

CPU

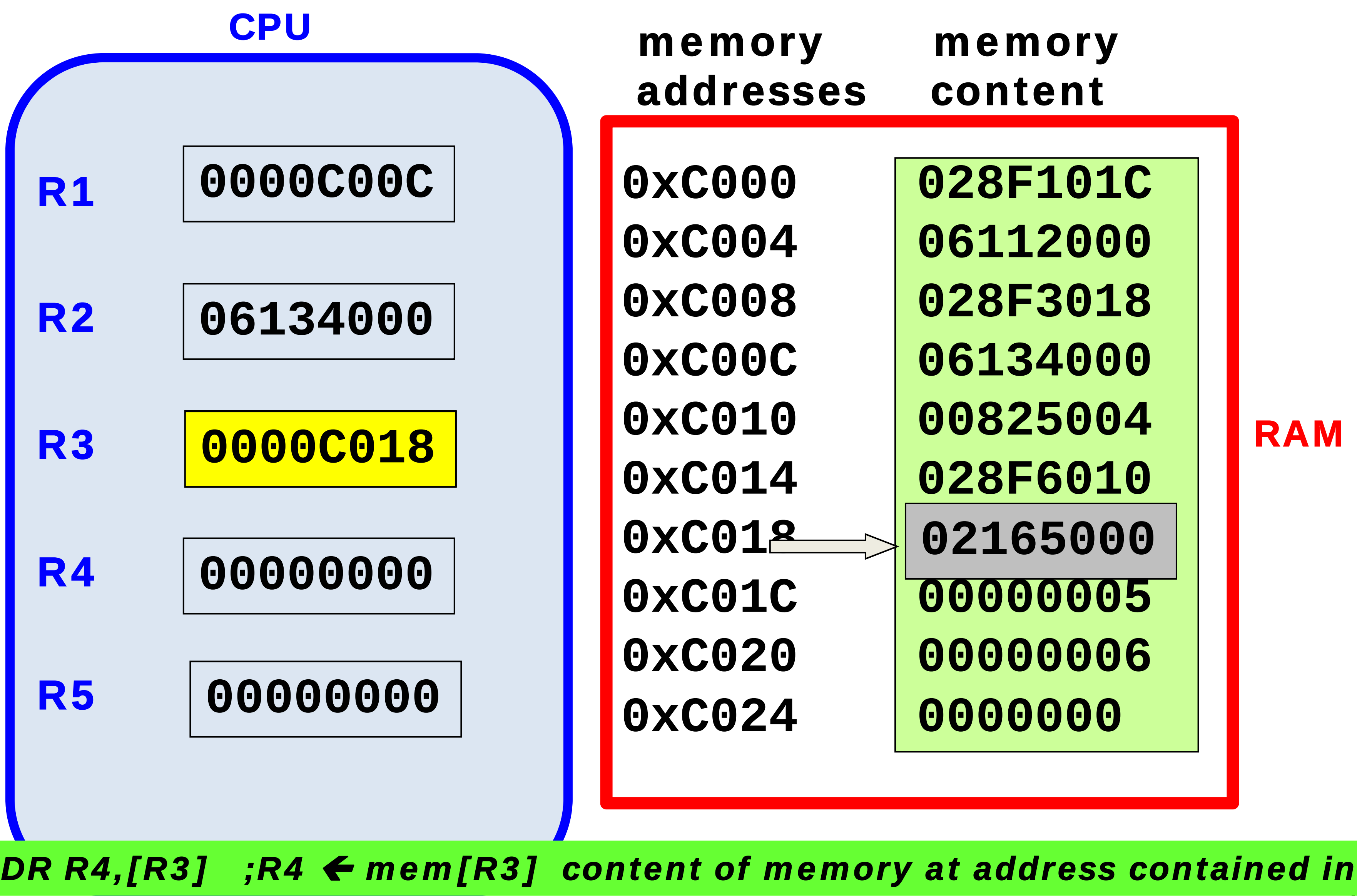
R1	0000C00C
R2	00000000
R3	0000C018
R4	00000000
R5	00000000

RAM

memory addresses	memory content
0xC000	028F101C
0xC004	06112000
0xC008	028F3018
0xC00C	06134000
0xC010	00825004
0xC014	028F6010
0xC018	02165000
0xC01C	00000005
0xC020	00000006
0xC024	0000000

LDR R2,[R1]   ;R2 ← mem[R1] content of memory at address contained in R1

Addressing and Pointers: by example



LDR R4,[R3] ;R4 ← mem[R3] content of memory at address contained in R3

Addressing and Pointers: by example

CPU

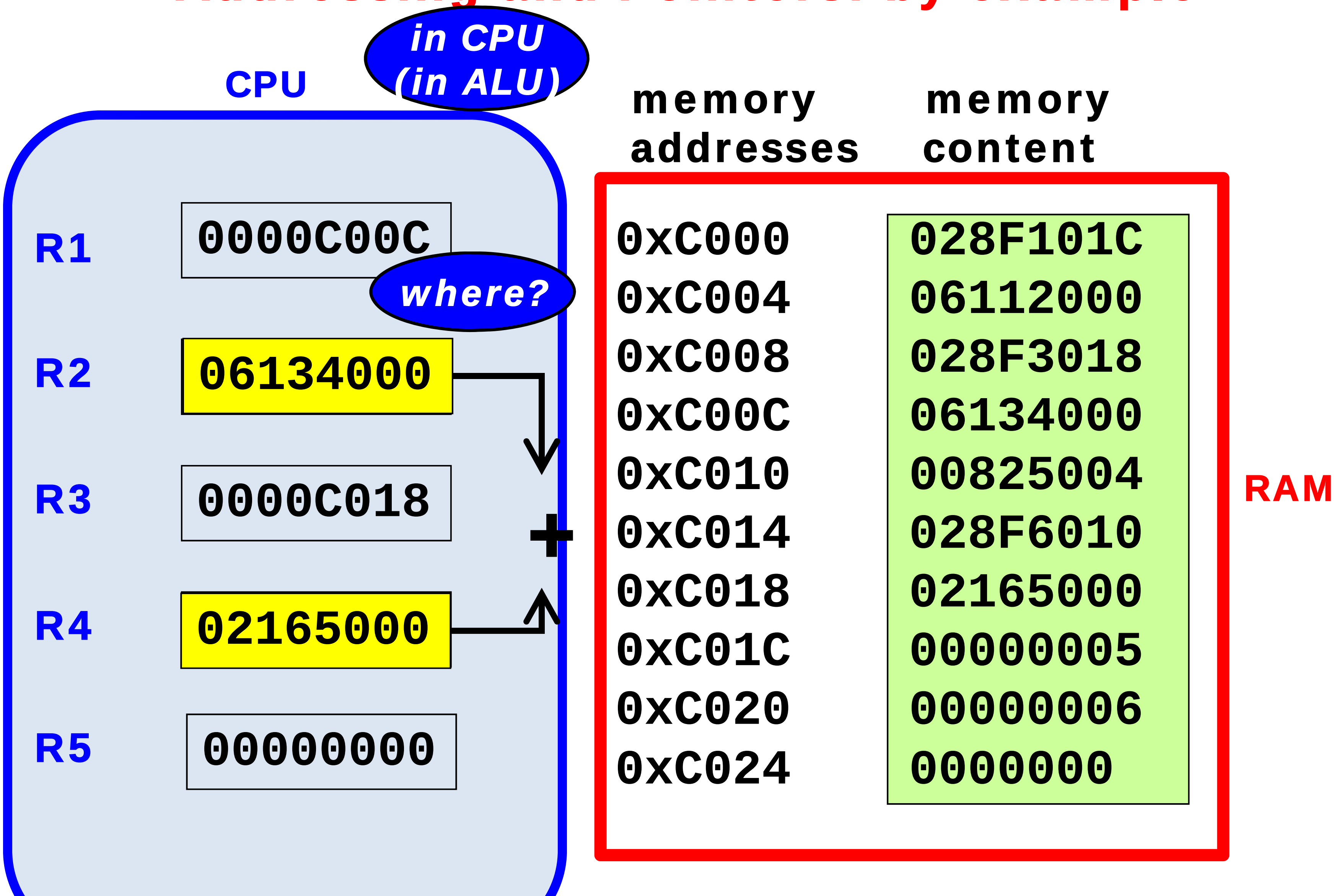
R1	0000C00C
R2	06134000
R3	0000C018
R4	02165000
R5	00000000

RAM

memory addresses	memory content
0xC000	028F101C
0xC004	06112000
0xC008	028F3018
0xC00C	06134000
0xC010	00825004
0xC014	028F6010
0xC018	02165000
0xC01C	00000005
0xC020	00000006
0xC024	0000000

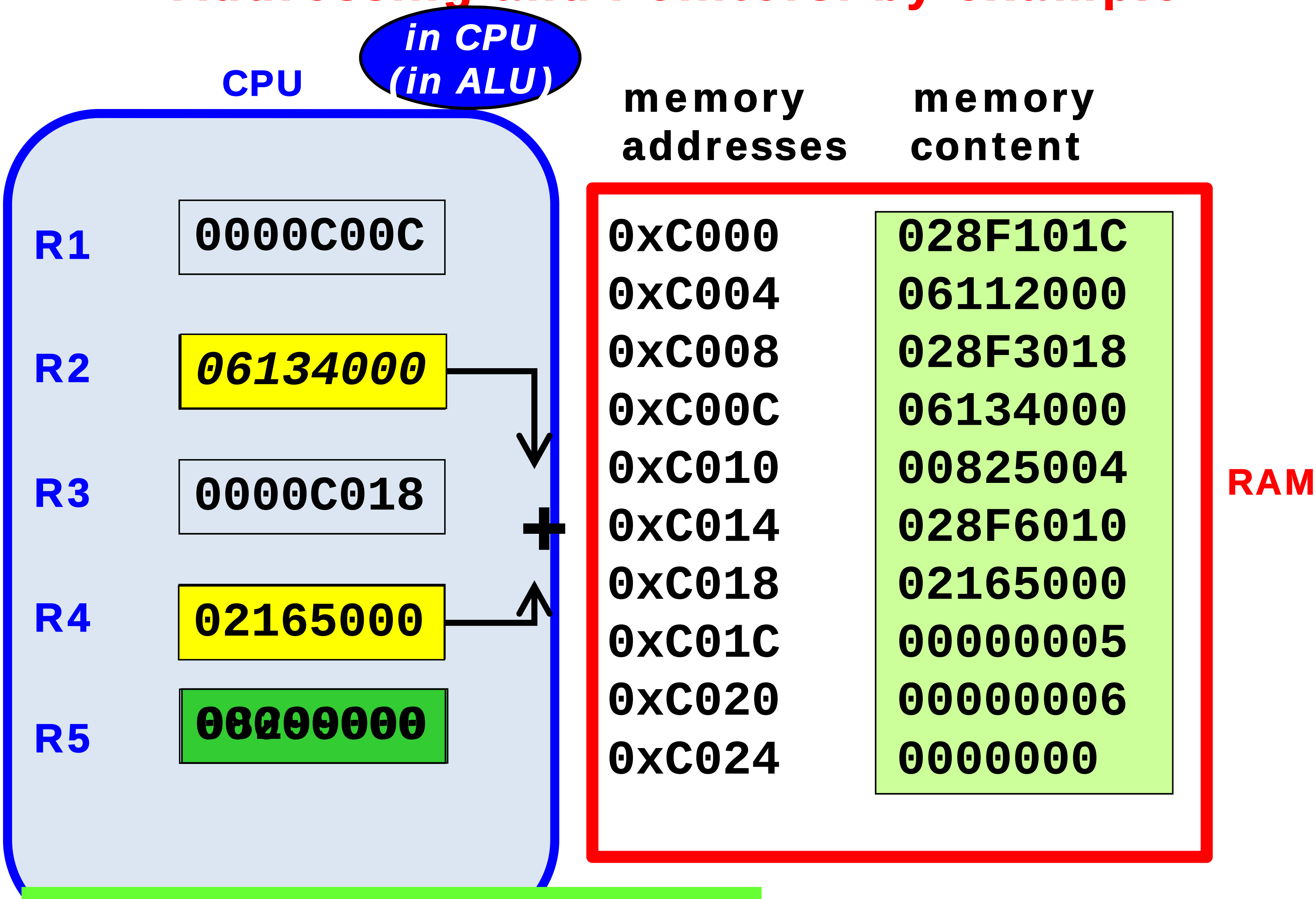
ADD R5,R2,R4 ; R5 = R2+R4

Addressing and Pointers: by example



ADD R5,R2,R4 ; R5 = R2+R4

Addressing and Pointers: by example



ADD R5,R2,R4 ; R5 = R2+R4

Another Example of addressing: an array

0x9000	00000001
0x9004	00000002
0x9008	00000003
0x900C	00000004
0x9010	00000005
0x9014	00000006
0x9018	00000007
0x901C	00000008
0x9020	00000009
0x9024	0000000A
0x9028	FFFF0123
0x902C	0012FF22
0x9030	00000000
0x9034	FFFFFFFF

Snapshot of memory starting at address 0x0000 9000

Assume an array of 10 elements has been declared and initialized to the numbers 1-10

array[0] = 1 ***at address 00009000***
array[1] = 2 ***at address 00009004***
array[2] = 3 ***at address 00009008***
array[3] = 4 ***at address 0000900C***
array[4] = 5 ***at address 00009010***
array[5] = 6 ***at address 00009014***
array[6] = 7 ***at address 00009018***
array[7] = 8 ***at address 0000901C***
array[8] = 9 ***at address 00009020***
array[9] = 10 ***at address 00009024***

Another Example of addressing: an array

0x9000	00000001
0x9004	00000002
0x9008	00000003
0x900C	00000004
0x9010	00000005
0x9014	00000006
0x9018	00000007
0x901C	00000008
0x9020	00000009
0x9024	0000000A
0x9028	FFFF0123
0x902C	0012FF22
0x9030	00000000
0x9034	FFFFFFFF

R1	00000000
R2	00000000

Look at the sequence of events to add 1 to the values of each element

```
for ( i=0,  i<10, i++)
        array[i] = array[i] + 1
```

Another Example of addressing: an array

0x9000	00000001
0x9004	00000002
0x9008	00000003
0x900C	00000004
0x9010	00000005
0x9014	00000006
0x9018	00000007
0x901C	00000008
0x9020	00000009
0x9024	0000000A
0x9028	FFFF0123
0x902C	0012FF22
0x9030	00000000
0x9034	FFFFFFFF

00009000	R1
00000001	R2

(1)Load the address of array into R1 (actually of the first element)

LDR R1,=array

(2)Now R1 is a *pointer* to the first element

(3)Load the content of the array at that position into R2

LDR R2,[R1]

Another Example of addressing: an array

0x9000	00000002
0x9004	00000002
0x9008	00000003
0x900C	00000004
0x9010	00000005
0x9014	00000006
0x9018	00000007
0x901C	00000008
0x9020	00000009
0x9024	0000000A
0x9028	FFFF0123
0x902C	0012FF22
0x9030	00000000
0x9034	FFFFFFFF

00009004 R1

00000002 R2

(4) Add 1 to the array value

ADD R2, R2, #1

(5) Store the updated value back in the correct array position

STR R2, [R1]

(6) Update the pointer to get to the next element of the array

ADD R1, R1, #4

Another Example of addressing: an array

0x9000	00000002
0x9004	00000003
0x9008	00000003
0x900C	00000004
0x9010	00000005
0x9014	00000006
0x9018	00000007
0x901C	00000008
0x9020	00000009
0x9024	0000000A
0x9028	FFFF0123
0x902C	0012FF22
0x9030	00000000
0x9034	FFFFFFFF

00009008 R1

00000003 R2

(1) Now R1 is a pointer to the 2nd element of the array

(2) Load the content of the array at that position into R2

LDR R2,[R1]

(4) Add 1 to the array value

ADD R2, R2, #1

(5) Store the updated value back

STR R2,[R1]

(6) Update the pointer to get to the next element of the array

ADD R1, R1, #4

Another Example of addressing: an array

0x9000	00000002
0x9004	00000003
0x9008	00000003
0x900C	00000004
0x9010	00000005
0x9014	00000006
0x9018	00000007
0x901C	00000008
0x9020	00000009
0x9024	0000000A
0x9028	FFFF0123
0x902C	0012FF22
0x9030	00000000
0x9034	FFFFFFFF

00009008	R1
00000003	R2

NOTES:

(1) Use R1 as a pointer to each element of the array in turn

(2) A pointer is simply a variable which contains the address of some other variable (even of another pointer)

(3) Example: P.O. Box numbers given in a newspaper ad

Another Example of addressing: an array

0x9000	00000002
0x9004	00000003
0x9008	00000003
0x900C	00000004
0x9010	00000005
0x9014	00000006
0x9018	00000007
0x901C	00000008
0x9020	00000009
0x9024	0000000A
0x9028	FFFF0123
0x902C	0012FF22
0x9030	00000000
0x9034	FFFFFFFF

Register	Value
R1	00009008
R2	00000003

(4) To load the content of a memory location into a register, one uses the special syntax of "index addressing", as in:

LDR R2, [R1]

(5) Same to store the updated value back: *STR R2, [R1]*

(6) One can do *pointer arithmetic!*

e.g. update the pointer to get to the next element of the array

ADD R1, R1, #4

Another Example of addressing: an array

0x9000	00000002
0x9004	00000003
0x9008	00000003
0x900C	00000004
0x9010	00000005
0x9014	00000006
0x9018	00000007
0x901C	00000008
0x9020	00000009
0x9024	0000000A
0x9028	FFFF0123
0x902C	0012FF22
0x9030	00000000
0x9034	FFFFFFFF

00009008	R1
00000003	R2

These are crucial concepts

They will be revisited often and in even more details

Think about it all!!!

Revisit pointers in C

```
int array[10];
int i;

for ( i=0,  i<10, i++)

      array[i] = array[i] + 1

}
```

```
int *p;
int array[10];
int i;

p = &array[0];

for( i=0; i < 10; i++)

         *p = *p +1;

         p++;

}
```