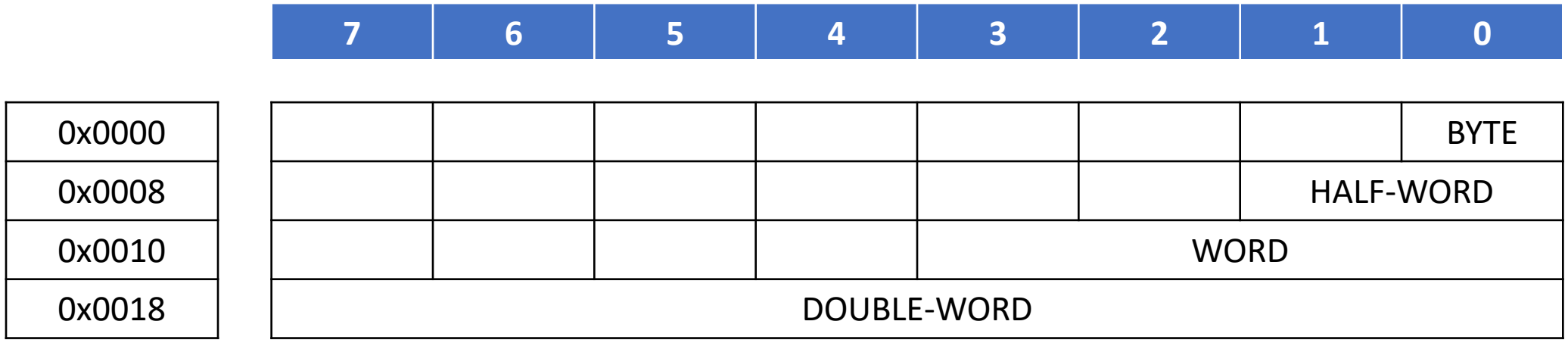
Memory addressing in assembly

Purpose

- How to access variables in assembly (byte, half-word, word, double-word)
- How to access an array

Computer memory organization

- Rows of 32 or 64 bits (4 or 8 bytes respectively)
- WinMIPS64: 64-bit Little-Endian Rows
- The memory allows access granularity equal to the smallest data type (byte)



Variables Declaration – C to assembly

- `char my_var;`
  - `my_var: .space 1`
- `int my_var;`
  - `my_var: .space 2`
  - `my_var: .space 4`
- `long int my_var;`
  - `my_var: .space 8`

Array Declaration – C to assembly

- `char my_var[10];`
  - `my_var: .space 10 ;10*1 bytes`
- `int my_var[10];`
  - `my_var: .space 20 ;10*2 bytes`
  - `my_var: .space 40 ;10*4 bytes`
- `long int my_var[10];`
  - `my_var: .space 80 ;10*8 bytes`

Constants – C to assembly

- `const char my_var[5]= {1, 2, 3, 4, 5};`
  - `my_var: .byte 1, 2, 3, 4, 5`
- `const char my_var = 5;`
  - `my_var: .byte 5`

Access to variables/costants – C to assembly

```
;const unsigned char my_var[5]= {1, 2, 3, 4, 5};
my_var: .byte 1, 2, 3, 4, 5 ;assuming my_var=0x0000
; read first byte
; tmp = *(my_var) ;
lbu  r7, my_var(r0)
; read third byte
; tmp = *(my_var + 2) ;
daddi r1, r0, 2
lbu  r7, my_var(r1)
```

	7	6	5	4	3	2	1	0
0x0000				5	4	3	2	1
0x0008								
0x0010								
0x0018								

Access to variables/costants – C to assembly

```
;const unsigned int my_var[5]= {1, 2, 3, 4, 5};
my_var: .word32 1, 2, 3, 4, 5 ;assuming my_var=0x0000
; read first word
; tmp = *(my_var) ;
lwu  r7, my_var(r0)
; read third word
; tmp = *(my_var + 2) ;
daddi r1, r0, 8
lwu  r7, my_var(r1)
```

	7	6	5	4	3	2	1	0
0x0000				2				1
0x0008				4				3
0x0010								5
0x0018								