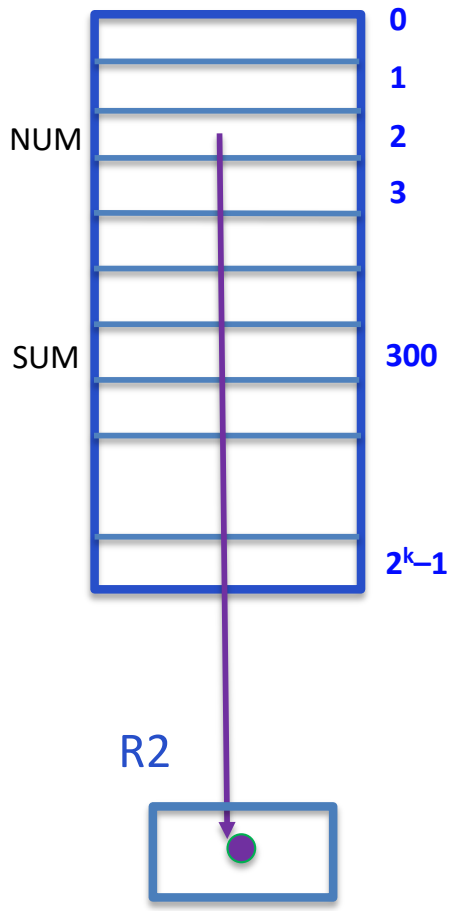
2.4 Operands and Effective Address

- Add R1, R2, R3 ; R1 ← R2 + R3
  - **Operands**: contents of R2 and R3; Add; Result in R1
- Store R2, SUM ; [SUM] ← R2
  - **Operand**: content of R2; Store content of R2 in memory address SUM
- Load R1, NUM ; R1 ← [NUM]
  - **Operand**: content of [NUM]; Load R1 with content of [NUM]
- **Effective Address EA**: where the operand is

[x]:=M[x] =content of memory location x

2.4.1 Direct or Absolute Naming

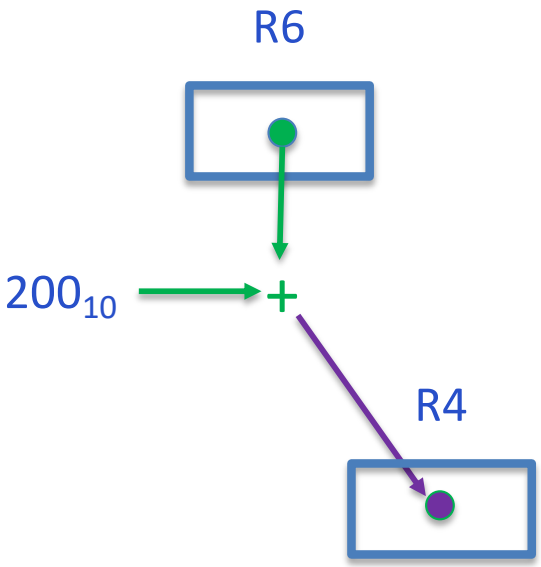
- Declaration in high-level language:
  - Integer NUM, SUM (user-defined symbols)
- Assembly level:
  - Specify locations by programmer or compiler
  - Load R2, NUM; (***Direct or Absolute***)
  - Constant or Variable is [NUM]



Register and Immediate

- Adding constant 200_{10} to a variable
  - Example:
  - Use ***Immediate*** with # sign
  - Add R4, R6, #200; (***Register***: R6)

Variable



2.4.2 Indirection and Pointers

- Load R2, (R5)
  - ***Register Indirect***: EA=[R5];
  - R5 contains the address of the operand and is a pointer
  - Example: Used as an index (pointer) to a 1-D array (a list)

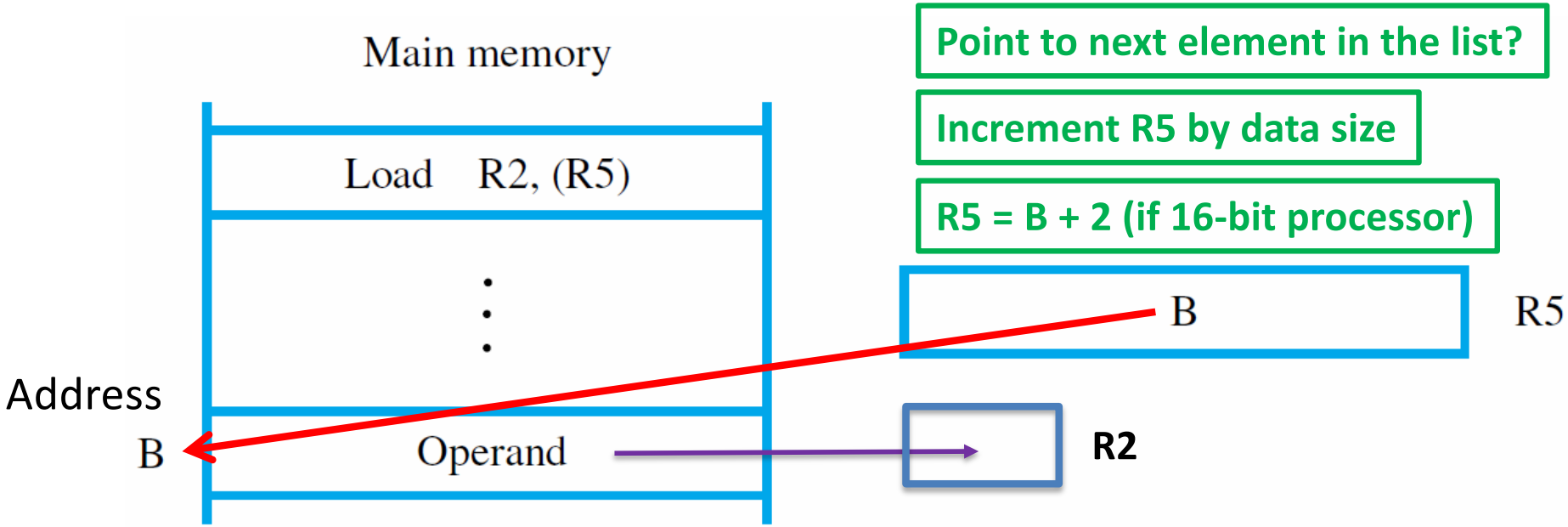
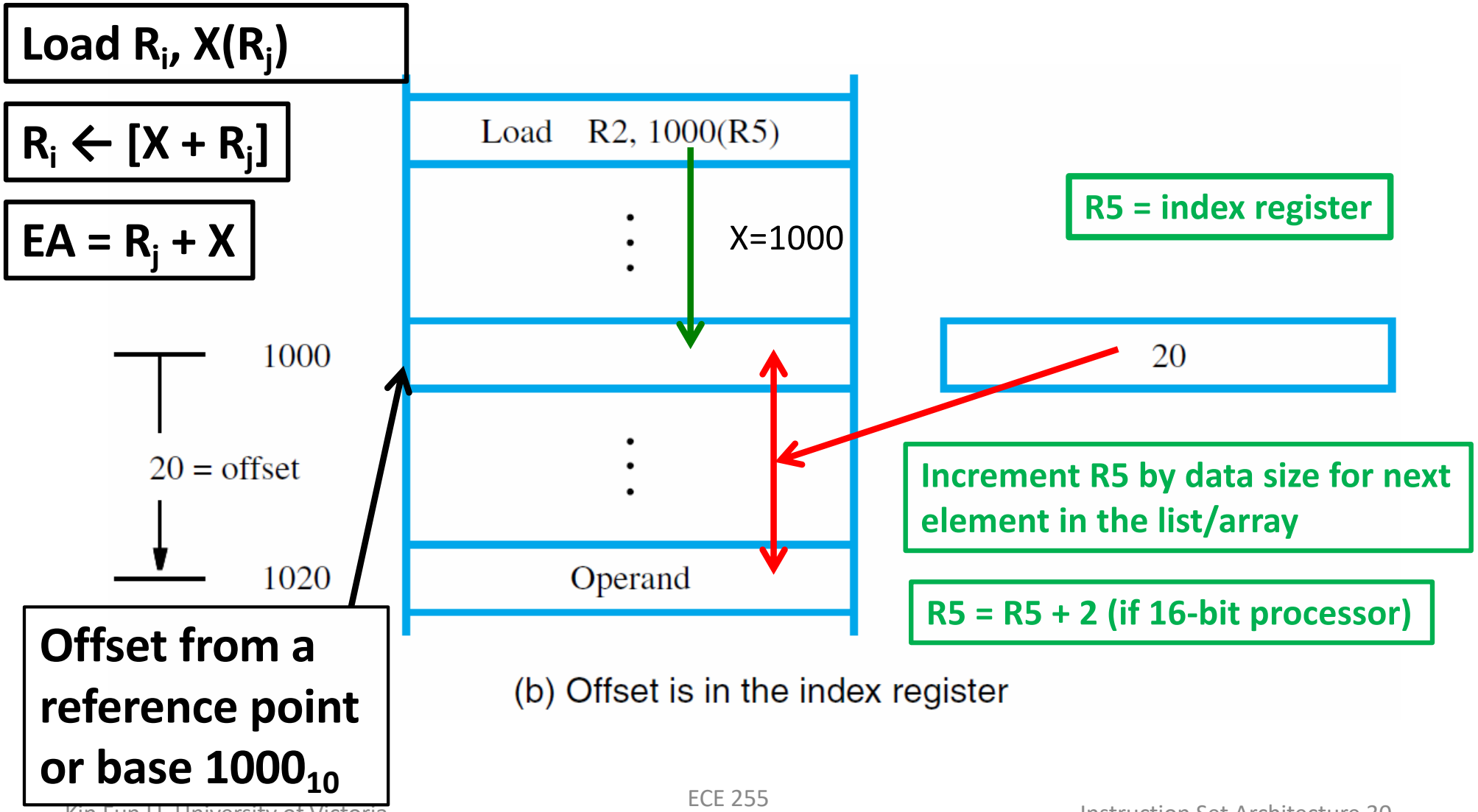


Fig. 2.9(b) Offset and Index



2.4.3 Indexing and Arrays

- Load R2, X(R5); e.g., Load R2, 1000(R5)
  - ***Index***: EA=[R5]+X ; base X, index R5
- Load R2, (R5,R6)
  - ***Base with Index***: EA=[R5]+[R6] ; base R5 index R6, or, index R6 base R5
- Assume: A 2 X 3 Matrix is in the Memory

$$\begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \end{bmatrix}$$

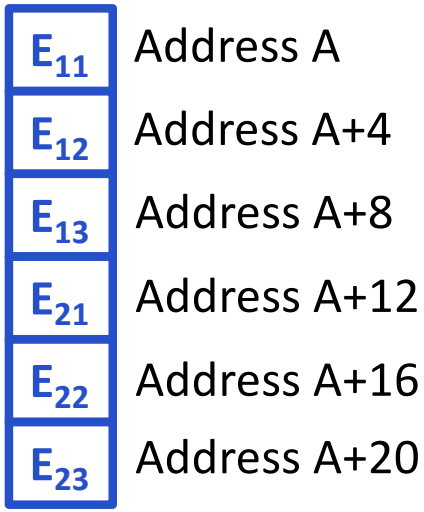
E_{11}	Address A
E_{12}	Address A+4
E_{13}	Address A+8
E_{21}	Address A+12
E_{22}	Address A+16
E_{23}	Address A+20

```
.align 4:
address
-100
-104
-108
```

Indexing Examples/Exercise

- 1: Use "X(R5)" to access matrix elements

  Load first row into R2, R3, R4



- 2: Use "(R5,R6)" to access matrix elements

  Load first row into R2, R3, R4