ECE3120: Computer Systems Chapter 4: Indexable Data Structures

Manjeera Jeedigunta
http://blogs.cae.tntech.edu/msjeedigun21
Email: msjeedigun21@tntech.edu
Tel: 931-372-6181, Prescott Hall 120

Indexable Data Structures (arrays)

- Vectors and matrices are indexable data structures.
- The first element of a vector is associated with the index 0 in order to facilitate the address calculation.
- Assemblers directives db, dc.b, fdb are used to define arrays of 8-bit elements.
- Assemblers directives dw, dc.w, and fdb are used to define arrays of 16-bit elements.
- directives ds,rmb, ds.b are used to reserve memory space for arrays with 8-bit element;
- directives ds.w and rmw are used to reserve memory space for arrays with 16-bit element;

.

Application perspective of Indexable data structures

- Searching an array / vector for a particular value
  - Sequential search
    - Elements not sorted
  - Binary search
    - Elements sorted in an order

Example 4.2 Write a program to find out if the array **vec_x** contains a value **key**. The array has 16-bit elements and is not sorted.

Solution:

-

Use the double accumulator to hold the key

- Use the index register X as a pointer to the array
  Use the index register Y to hold the loop count
  Need to compare **key** with every array element because it is not sorted

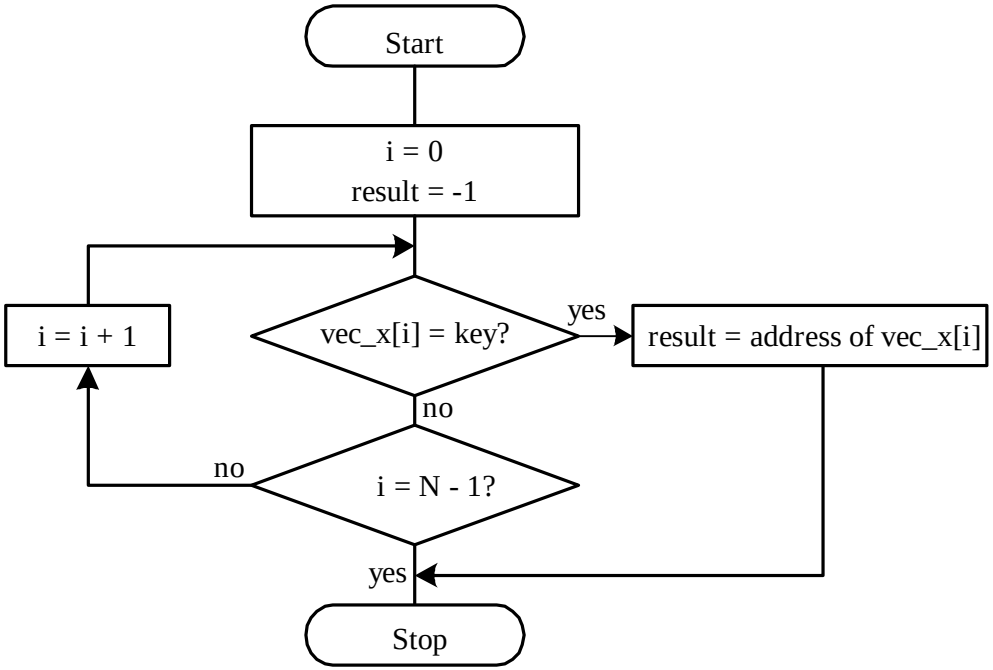

Figure 4.3 Flowchart for sequential search

Sequential Search

```
N           equ   10              ;array count
Notfound    equ   -1
Key         equ   190             ;define the searching key
Vec_x       dw    13,15,320,37,190,300,650,777,555,444
            org   $1000
Result      rmw   1               ;reserve a word for the result
            org   $1500           ;start program
            ldy   #N              ;set up loop count
            ldd   #notfound
            std   result          ;initiliaze search result
            ldd   #key
            ldx   #vec_x          ;place the starting address of the array
```

Initialization

Sequential Search

```
Loop    cpd     2,X+    ;compare the key with current array
                        ;element and move to the next element
        beq     found
        dbne    Y,loop  ;checking the limits for the loop
        bra     done
Found   dex             ;need to restore the value of X to
        dex             ; point to the matched element
        stx     result
Done    swi
        end
```

Binary Search

Step 1

Initialize variables max and min to n -1 and 0, respectively.

Step 2

If max < min, then stop. No element matches the key.

Step 3

Let mean = (max + min)/2

Step 4

If key = arr[mean], then key is found in the array, exit.

Step 5

If key < arr[mean], then set max to mean - 1 and go to step 2.

Step 6

If key > arr[mean], then set min to mean + 1 and go to step 2.

Binary search example

Step 1
Max=N-1;Min=0
Step 2
Max<Min à stop
Step 3
Mean=max+min/2
Step 4
Key=arr[mean] à found
Step 5
Key<arr[mean]
Max=mean-1;goto step2
Step 6
Key>arr[mean]
Min=mean+1;goto setp2

Array = 1,2,**4**,5,8,9,25,29,32

Key = 4

N = 9

Max=N-1=8

Min=0

Mean=4 ; Ist iteration

Arr[4]=8 à !=key à notfound

Key<arr[4] à Max=4-1=3

Mean=1 ; 2nd iteration

Arr[1]=2 à !=key à notfound

Key>arry[1] à min=1+1=2

Mean=2 ; 3rd iteration

Arr[2]=key=found

Example 4.3 Write a program to implement the binary search algorithm and also a sequence of instructions to test it.

Solution:

```
n          equ   30            ; array count
key        equ   69            ; key to be searched
arr        db    1,3,6,9,11,20,30,45,48,60
           db    61,63,64,65,67,69,72,74,76,79
           db    80,83,85,88,90,110,113,114,120,123
           org   $1000
max        rmb   1             ; maximum index value for
comparison
min        rmb   1             ; minimum index value for comparison
mean       rmb   1             ; the average of max and min
result     rmb   1             ; search result
           org   $1500
           clra
           staa  min           ; initialize min to 0
           staa  result        ; initialize result to 0
           ldaa  #n-1
           staa  max           ; initialize max to n-1
           ldx   #arr          ; use X as the pointer to the array
```

Step 1

Max=N-1;Min=0

Step 2

Max<Min à stop

Step 3

Mean=max+min/2

Step 4

Key=arr[mean] à found

Step 5

Key<arr[mean]

Max=mean-1;goto step2

Step 6

Key>arr[mean]

Min=mean+1;goto setp2

```
loop        ldab    min
            cmpb    max
            lbhi    notfound
            addb    max         ; compute mean
            lsrb                ;          “
            stab    mean        ; save mean
            ldaa    b,x         ; get a copy of the element
                                ;arr[mean]
            cmpa    #key
            beq     found
            bhi     search_lo   ;key<arr[mean]
            ldaa    mean        ;key>arr[mean]
            inca
            staa    min         ; place mean+1 in min to continue
            bra     loop
search_lo   ldaa    mean
            deca
            staa    max
            bra     loop
found       ldaa    #1
            staa    result
notfound    swi
            end
```

Step 1

Max=N-1;Min=0

Step 2

Max<Min à stop

Step 3

Mean=max+min/2

Step 4

Key=arr[mean] à found

Step 5

Key<arr[mean]

Max=mean-1;goto step2

Step 6

Key>arr[mean]

Min=mean+1;goto setp2

Next…

- Strings