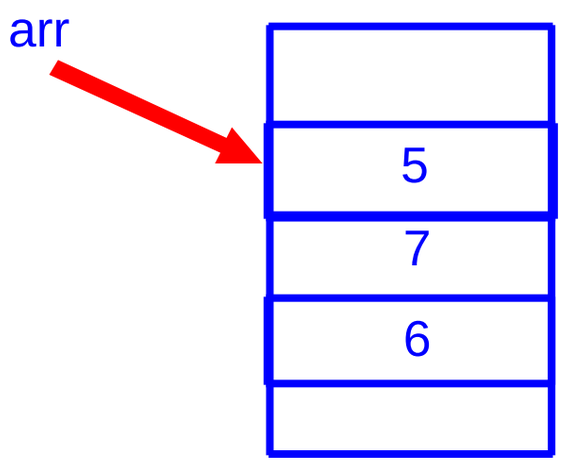
Australian National University

Arrays and Structures

Eric McCreath

Arrays

- Arrays translate into machine code very easily by having a pointer to the start of the array and and knowing the size of each element.
- The address of a particular element can be calculated by multiplying the size of elements by the index of the element you are after, and then adding this to the memory address of the start of the array.



int arr[] = {5,7,6};

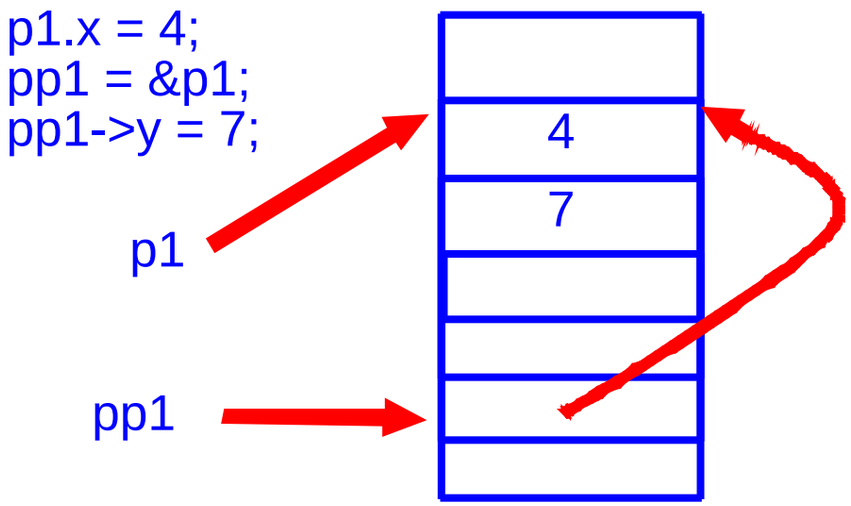
- Most ISA will have instructions that makes it easy to index elements of an array.

Arrays and rPeANUt

```
;   int sum;
;   int array[3];
0x0100 : load #5 R0       ; array[0] = 5;
         load #array R1
         store R0 R1
         load #7 R0       ; array[1] = 7;
         load #array R1
         load #1 R2
         add R1 R2 R1
         store R0 R1
         load #9 R0       ; array[2] = 9;
         load #array R1
         load #2 R2
         add R1 R2 R1
         store R0 R1
         load #3 R3       ; sum = 3 + array[2];
         load #array R1
         load #2 R2
         add R1 R2 R1
         load R1 R4
         add R3 R4 R0
         store R0 sumvar
         halt
array : block 3
sumvar : block 1
```

Structures

- Structures are just templates of how memory is laid out for a set of data elements.
- This can be simply implemented by using a pointer to the location of the structure in memory and then offsets to the element that make up the structure.




```
struct xypoint {
    int x;
    int y;
};

struct xypoint p1;
struct xypoint *pp1;
```

Structures

```
;struct xypoint {
;   int x;      // offset of #0 in the struct
;   int y;      // offset of #1 in the struct
;};
;struct xypoint p1;
;struct xypoint *pp1;
0x0100 : load #p1 R1     ; p1.x = 4;
         load #0 R2
         add R1 R2 R3
         load #4 R0
         store R0 R3

         load #p1 R1     ; pp1 = &p1;
         store R1 pp1

         load pp1 R1     ; pp1->y = 7;
         load #1 R2
         add R1 R2 R3
         load #7 R0
         store R0 R3

         halt

p1 : block 2 ; sizeof(struct xypoint)
pp1 : block 1
```

Exercises

- Implement the below in rPeANUt.

```
struct list {
   int nums[10];
   int size;
}
void add(struct list *lp, int v) {
   lp->nums[lp->size] = v;
   lp->size++;
}
struct list l1;
int main() {
   l1.size = 0;
   add(&l1, 7);
   add(&l1, 5);
   add(&l1, 4);
}
```

- This involves: structs, arrays, and functions calls. So take care and check each part of this exercise is working correctly.
- What is the problem with this list implementation?