5.12 Arrays and strings

Arrays

In C, an ***array*** is a variable consisting of a sequence of ***elements***. Ex: int x[4] defines 4 elements, accessed as x[0], x[1], x[2], array's elements are stored sequentially in memory, with a starting address known as the ***base address*** (or just base). So if then x[0] is at 5000, x[1] 5004, x[2] 5008, and x[3] 5012 (recalling word addresses increment by 4).

In assembly, accessing element x[i] requires calculating the element's address as: base + 4*i. Ex: If x's base is 5000, then x[5000 + 4*2 = 5008.

PARTICIPATION ACTIVITY 5.12.1: Declaring an array, and calculating an element's address.

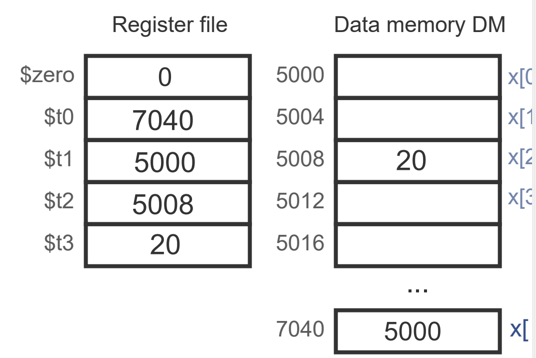
Start 2x speed

```
int x[4];

x[2] = 20;
```

```
addi $t0, $zero, 7040  # x var's addr
addi $t1, $zero, 5000  # x's base
sw $t1, 0($t0)         # Store base in x

lw $t1, 0($t0)         # Get x's base
addi $t2, $t1, 8       # x[2]'s address
addi $t3, $zero, 20    # $t3 = 20
sw $t3, 0($t2)         # x[2] = 20
```



PARTICIPATION ACTIVITY 5.12.2: Arrays in assembly.

Consider the above animation.

1) int x[4] defines an array of how many elements?

- 3
- 4

2) x's base address is _____ .

- 5000
- 7040

3) x's base address is stored at address _____ .

- 5000
- 7040

4) Which instruction is used to get x's base address, to begin calculating an element's address?

- `lw $t1, 0($t0)`
- `sw $t1, 0($t0)`

5) The calculation for x[1] would add what to the base address 5000?

- 1
- 4
- 8

6) At what address is x[0]?

- 5000
- 5001
- 5004

7) Given another array declared as int z[300] with base address 6000, at what address is element z[100]?

- 6100
- 6400
- 7200

8) Which instructions write the address of x[1] into $t1?

- ```
  addi $t6, $zero, 7040
  lw $t0, 0($t6)
  addi $t1, $t0, 4
  ```
- ```
  addi $t6, $zero, 7040
  lw $t0, 0($t6)
  addi $t1, $t0, 1
  ```

9) Assuming x[1]'s address is in $t1, which instruction writes $t6 with x[1]'s value?

- `add $t6, $t1, $zero`
- `lw $t6, 0($t1)`

Arrays and loops

One benefit of an array versus one variable per element is efficient handling in loops, as shown below.

Figure 5.12.1: Array example in C.

Assume int x[51] and int i.

```
i = 0;
while (i <= 50) {
   x[i] = i * i;
   i = i + 1;
}
// x will be 0, 1, 4, 9, ..., 2500
```

Figure 5.12.2: Above array example in assembly.

Assume: $t0 has x's base of 5000, $t1 has 50, and $t2 has 4.

```
Line   #
1          addi $t3, $zero, 0  # i = 0;
2      While:
3          bgt $t3, $t1, After  # while (i <= 50)
4          mul $t4, $t3, $t2   # $t4 = i * 4
5          add $t4, $t0, $t4   # $t4 = x's base + i*4
6          mul $t5, $t3, $t3   # $t5 = i * i
7          sw $t5, 0($t4)      # x[i] = i * i;
8          addi $t3, $t3, 1    # i = i + 1;
9          j While
10
11     After:
```

PARTICIPATION ACTIVITY 5.12.3: Arrays and loops.

Consider the figure above showing assembly.

1) In the first iteration, i ($t3) is 0. What is $t4 after line 4 executes?

- 0
- 4

2) In the first iteration, what is $t4 after line 5 executes?

- 0
- 5000
- 5004

3) In the second iteration, what element is being written?

- x[0]
- x[1]
- x[2]

4) In the second iteration, what address is calculated in line 5?

- 5000
- 5004
- 5008

5) In the last iteration, i will be 50. What address will the sw instruction store into?

- 2500
- 50
- 200
- 5200

6)

Suppose the array was int x[100] rather than int x[50]. How many of the shown loop instructions need to be modified?

- 0
- 1
- 2

CHALLENGE ACTIVITY 5.12.1: Arrays in assembly.

Start

Convert the C to assembly. x's base address is 5000. Store x's base address at DM[7056].

```
int x[4];
```

addi ▼	$t0 ▼,	$zero ▼,	0
addi ▼	$t0 ▼,	$zero ▼,	0
addi ▼	$t0 ▼,	$zero ▼,	0

Registers

$zero	0
$t0	0
$t1	0

Data memory

5000	0
7056	0

1 2 3 4 5

Check Next

Strings

In C, a ***string*** is an array of characters. Each character is stored as a number, being the character's ASCII value. The last ele string is always the ***null character*** '\0', whose ASCII value is 0.

PARTICIPATION ACTIVITY 5.12.4: A C string is an array of ASCII values, ending with 0.

Start 2x speed

```
char s[3] = "Hi";


i = 0;
while (s[i] != '\0') {
  printf("%c", s[i]);
  i = i + 1;
}
```

Data memory D

5100	72	s[0]	*H*
5104	105	s[1]	*i*
5108	0	s[2]	*"null" character '\0'*
5012			
	...		
7060	5100	s[ ]	

A programmer can leave the array size blank, as in `char myStr[] = "Hi";`. The compiler will create an array with the app of elements, in this case 3, with the last element being the null character.

A character is 8 bits (one byte), while a memory word is 32 bits. Thus, in MIPS, a character array is stored with four charact with each successive element having an address incremented by 1 (not 4). MIPS has instructions lb (load byte) and sb (stc access bytes within a word. However, for simplicity of introduction, MIPSzy only has lw (load word) and sw (store word), ar four characters per word is not discussed here.

PARTICIPATION ACTIVITY 5.12.5: C strings.

1) For char s[4] = "Hey", what character is s[1]?

Check Show answer

2) For char s[] = "Hiya", a compiler creates an array with how many elements?

Check Show answer

3) For char s[] = "a0b1", what is the value stored in s[1]? (Note: Use an ASCII lookup table on the web).

Check Show answer

4) For char s[] = "0123", what is the ASCII value of s[4]?

Check Show answer

5) char s[] = "1234567" requires 8 words in MIPSzy but only _____ words in MIPS.

Check Show answer

Provide feedback on this section