Lecture 20

Arrays in IA-32

CPSC 275
Introduction to Computer Systems

Basic Data Types in IA-32

- Integral
  - Stored and operated on in general (integer) registers
  - Signed vs. unsigned depends on instructions used

Intel	ASM	Bytes	C
byte	`b`	1	`[unsigned] char`
word	`w`	2	`[unsigned] short`
double word	`l`	4	`[unsigned] int`

- Floating point (later)
  - Stored & operated on in floating point registers

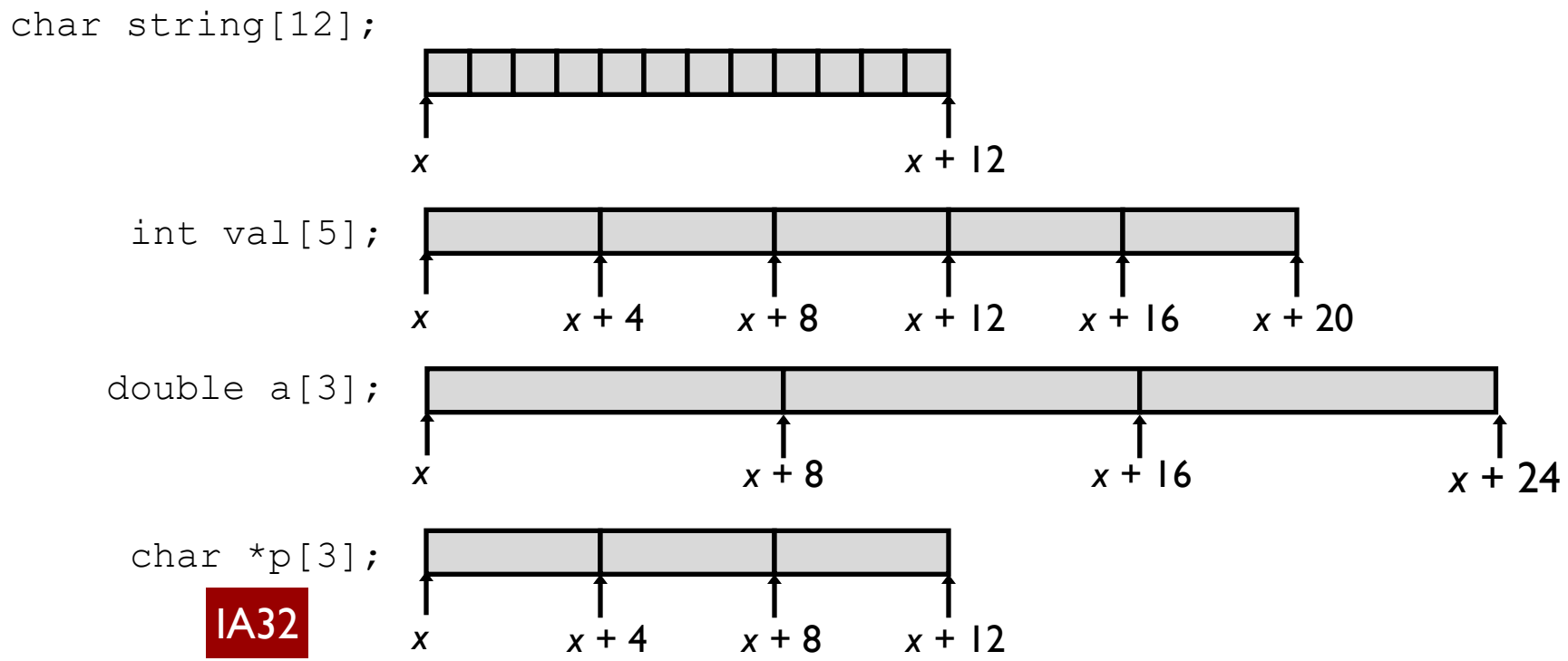
Intel	ASM	Bytes	C
single	`s`	4	`float`
double	`l`	8	`double`

Array Allocation

- Basic principle

T **`A[`***L***`];`**

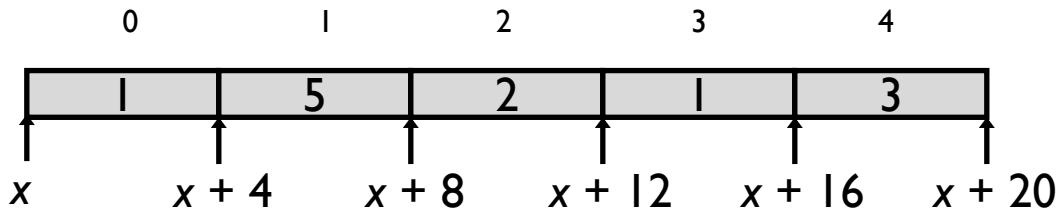
  - Array of data type *T* and length *L*
  - Contiguously allocated region of *L* * **`sizeof`**`(`*T*`)` bytes



IA32

Array Access

```
int val[5];
```



Reference	Type	Value
`val[4]`	`int`	3
`val`	`int *`	*x*
`val+1`	`int *`	*x* + 4
`&val[2]`	`int *`	*x* + 8
`val[5]`	`int`	??
`val + 5`	`int *`	*x* + 20
`*(val+1)`	`int`	5
`val + i`	`int *`	*x* + 4*i*

Array Operations in IA-32

Assume:

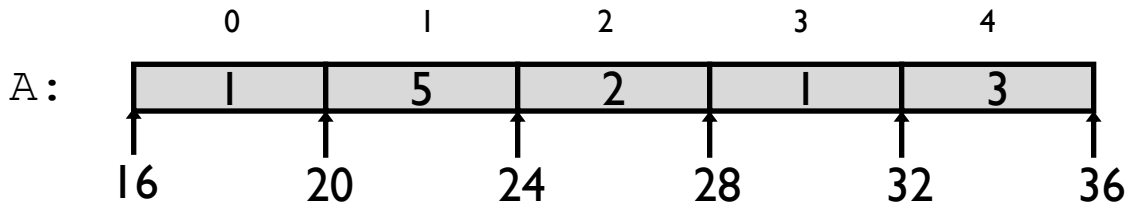
`%edx` stores the address of `arr`,

`%ecx` stores the value `i`

`%eax` represents some variable `x`.

Operation	**Type**	**Assembly**
`x = arr`	`int *`	`movl %edx, %eax`
`x = arr[0]`	`int`	`movl (%edx), %eax`
`x = arr[i]`	`int`	`movl (%edx,%ecx,4), %eax`
`x = &arr[3]`	`int *`	`leal 12(%edx), %eax`
`x = arr+3`	`int *`	`leal 12(%edx), %eax`
`x = *(arr+3)`	`int`	`movl 12(%edx), %eax`

Example




```
int get_digit(int z[], int n) {
  return z[n];
}
```

IA-32:

```
movl (%edx,%ecx,4),%eax
```

- Suppose that register `%edx` contains starting address of array
- Register `%ecx` contains array index
- Desired digit at ?

  `%edx + 4*%ecx`
- Memory reference?

  `(%edx,%ecx,4)`

Array Loop Example

```
#define ZLEN 5
void zincr(int z[]) {
  int i;

  for (i = 0; i < ZLEN; i++)
     z[i]++;
}
```

```
                                # %edx = z
  movl  $0, %ecx                # %ecx = i = 0
.L4:                            # loop:
  incl  (%edx,%ecx,4)           #   z[i]++
  incl  %ecx                    #   i++
  cmpl  $5, %ecx                #   compare i with 5
  jne   .L4                     #   if !=, goto loop
```

Pointer Loop Example

```
#define ZLEN 5
void zincr_p(int z[]) {
  int *zend = z + ZLEN;

  do {
     (*z++)++;
  } while (z != zend);
}
```

```
                                # %edx = z = p (char pointer)
  movl  $0, %ecx                #   i = 0
.L8:                            # loop:
  incl  (%edx,%ecx)             #   increment *(p+i)
  addl  $4, %ecx                #   i += 4
  cmpl  $20, %ecx               #   has reached the end?
  jne   .L8                     #   if !=, goto loop
```