Memory in assembly

1/10/25

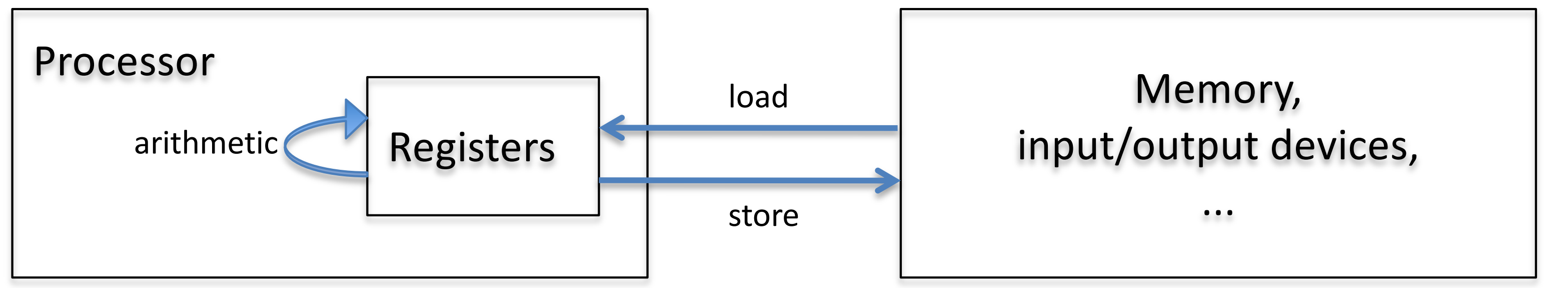
Administrivia

- HW 1 (ASCII art in assembly) due Wednesday

- Candidate’s research talk today at 4:15pm in SMC A202 (cookies at 3:45)

  Extra credit if you email me a writeup (or several)

Recall: Assembly instruction cartoon



Memory

- Big array of numbers
- Each byte (8 bit value) gets an address

Memory

- Big array of numbers
- Each byte (8 bit value) gets an address
- Complication: Objects are different size
  - In MARS, integers are 4 bytes long
  - So are memory addresses

Loading and storing integers

- To store an int from a register to memory:

```
sw    reg, address        #“store word”
```

- To load an int from memory to a register:

```
lw    reg, address        #“load word”
```

- For both, address is

```
(register)             #use register value
imm(register)          #use imm + register value
```

$\uparrow$ i.e. an integer

Which of the following is syntactically valid?

A. lw $t0, ($t0)

B. sw $t1, -4($s1)

C. sw 0($a0), $t3

D. lw ($t5), $a5

E. Not exactly one of the above

Which of the following is syntactically valid?

A. lw $t0, ($t0)

B. sw $t1, -4($s1)

C. sw 0($a0), $t3

D. lw ($t5), $a5

E. Not exactly one of the above (A & B)

Two kinds of memory errors

- Non-aligned memory address:

```
lw      $a0, 11($zero)
```

Two kinds of memory errors

- Non-aligned memory address:

```
lw      $a0, 11($zero)
```

- Illegal memory address (address out of range):

```
lw      $a0, 16($zero)
```

Reserving memory inside a program

```
        .data
        .align 2
var:    .word 10, -1
var2:   .space 4

        .text
        la  $a0, var
        lw $t0, ($a0)
        lw $t1, 4($a0)
```

Reserving memory inside a program

```
       .data
       .align 2
var:   .word 10, -1
var2:  .space 4

       .text
       la  $a0, var
       lw $t0, ($a0)
       lw $t1, 4($a0)
```

switch to writing in data segment (where variables live)

switch to writing in text segment (where code lives)

Reserving memory inside a program

```
       .data
       .align 2
var:   .word 10, -1
var2:  .space 4

       .text
       la  $a0, var
       lw $t0, ($a0)
       lw $t1, 4($a0)
```

skip as needed so the address of the next object is multiple of $2^2 = 4$

Reserving memory inside a program

```
        .data
        .align 2
var:    .word 10, -1
var2:   .space 4

        .text
        la  $a0, var
        lw $t0, ($a0)
        lw $t1, 4($a0)
```

store integer (word) 10 at location whose address is labeled “var” and integer -1 four bytes later

reserve 4 bytes of space at location whose address is labeled “var2”

Reserving memory inside a program

```
        .data
        .align 2
var:    .word 10, -1
var2:   .space 4

        .text
        la  $a0, var        #put address of var into $a0
        lw $t0, ($a0)       #loads the 10
        lw $t1, 4($a0)      #loads the -1
```

Arrays in assembly

- Contents stored in contiguous memory, one cell after another
  - Each cell is sized for the object being stored
  - Address of i^th cell:

    address of 0^th + i * object_size

Arrays in assembly

- Contents stored in contiguous memory, one cell after another
  - Each cell is sized for the object being stored
  - Address of i^{th} cell:

    address of 0^{th} + i * object_size

- No memory protection or notion of array's length

Easy way to multiply by a power of 2

- sll ("shift left logical") instruction:

```
sll     $t0, $t1, 1     #$t0 = $t1 * 2
sll     $t0, $t1, 2     #$t0 = $t1 * 4     (2^2 = 4)
sll     $t0, $t1, 3     #$t0 = $t1 * 8     (2^3 = 8)
```

- Last number is # zeros to add to end of binary representation of $t1

Which of the following loads the value of array[i+3] into $a0?

($t0 has beginning of array (of ints); $t1 has i)

A.
```
lw   $a0, 12($t0)
```
B.
```
sll  $t2, $t1, 2
addi $t2, $t2, 3
add  $t2, $t2, $t0
lw   $a0, ($t2)
```
C.
```
addi $t2, $t1, 3
sll  $t2, $t2, 2
add  $t2, $t2, $t0
lw   $a0, ($t2)
```
D.
```
sll  $t2, $t1, 2
add  $t3, $t2, $t0
lw   $a0, 12($t3)
```
E. Not exactly one of the above

Which of the following loads the value of array[i+3] into $a0?

($t0 has beginning of array (of ints); $t1 has i)

A.
```
lw    $a0, 12($t0)
```
B.
```
sll   $t2, $t1, 2
addi  $t2, $t2, 3
add   $t2, $t2, $t0
lw    $a0, ($t2)
```
C.
```
addi  $t2, $t1, 3
sll   $t2, $t2, 2
add   $t2, $t2, $t0
lw    $a0, ($t2)
```
D.
```
sll   $t2, $t1, 2
add   $t3, $t2, $t0
lw    $a0, 12($t3)
```
E. Not exactly one of the above (C & D)

What about strings?

- Array of chars (one byte each)
  - End marked with 0 (not ‘0’)

- Access individual chars with
  - lbu register, address   #“load byte unsigned”
  - sb   register, address   #“store byte”

Which of the following lines of code is incorrect for a loop that prints a string one char at a time?

```
#assume t0 has the address of the string
        lbu     $a0, ($t0)              #A
loop:   beq     $a0, $zero, exit        #B
        addi    $v0, $zero, 11
        syscall
        addi    $t0, $t0, 1             #C
        b       loop
exit:   ...     #exit the program (or whatever)
```

#D = None, it works   E = Something else

Which of the following lines of code is incorrect for a loop that prints a string one char at a time?

```
#assume t0 has the address of the string
        lbu     $a0, ($t0)              #A
loop:   beq     $a0, $zero, exit        #B
        addi    $v0, $zero, 11
        syscall
        addi    $t0, $t0, 1             #C
        b       loop
exit:   ...     #exit the program (or whatever)
```

#D = None, it works   E = Something else

(Need to read new char after increment)

Other important string information

- Better way to print a string: syscall 4

- To store a string into the data segment:

```
        .data
str:    .asciiz    “hello world”
```