S'More Instructions*

MIPS instructions for moving data between memory and the registers are introduced, as are branching instructions for *if* and looping constructions. There is an example, and an explanation of two addressing modes.

© 1995 LS All Rights Reserved

*Cover a Graham cracker with a marshmallow and chocolate.

Memory

Where programs and data reside when they are not being operated upon.

Every memory location (storage cell) has an address
Memories are byte addressed, from 0

Byte = 8 bits (Character)

Word = 32 bits

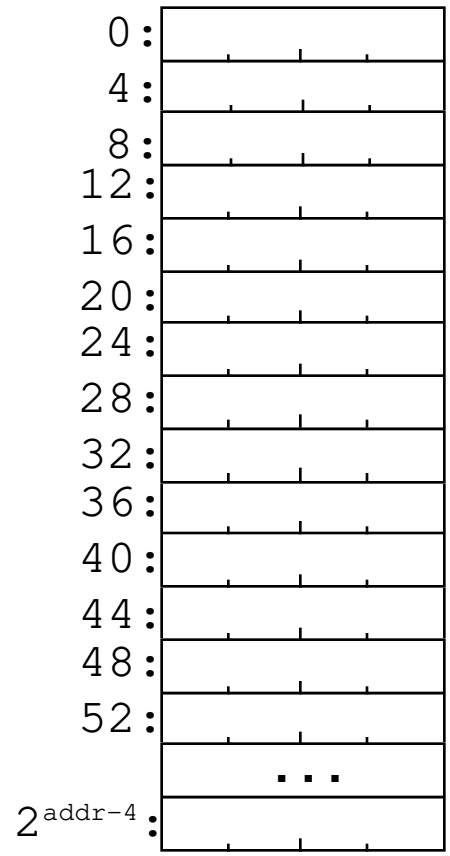
Half word = 16 bits

Double word = 64 bits

Typical computers use 30 bits of address: 2^{30}-1
Byte order: which byte of a word is 0^{th}

"Big-endian" -- 0^{th} byte at MSB end | 0 | 1 | 2 | 3 |

"Little-endian" -- 0^{th} byte a LSB end | 3 | 2 | 1 | 0 |



© 1995 LS All Rights Reserved

Moving Data To/From Memory

- In Load/Store Architectures, data must be moved from memory to registers before it can be used in computation.
- Unit of transfer -- word, half word, byte, and often others
- Base/Offset Addressing, or Base Addressing for short
  - Add address to contents of register forming *effective address*

```
lw  $8, 4($19)    # reg 8 = wd at address 4+$19
sw  $5,12($19)    # wd at address 12+$19 = reg 5
```

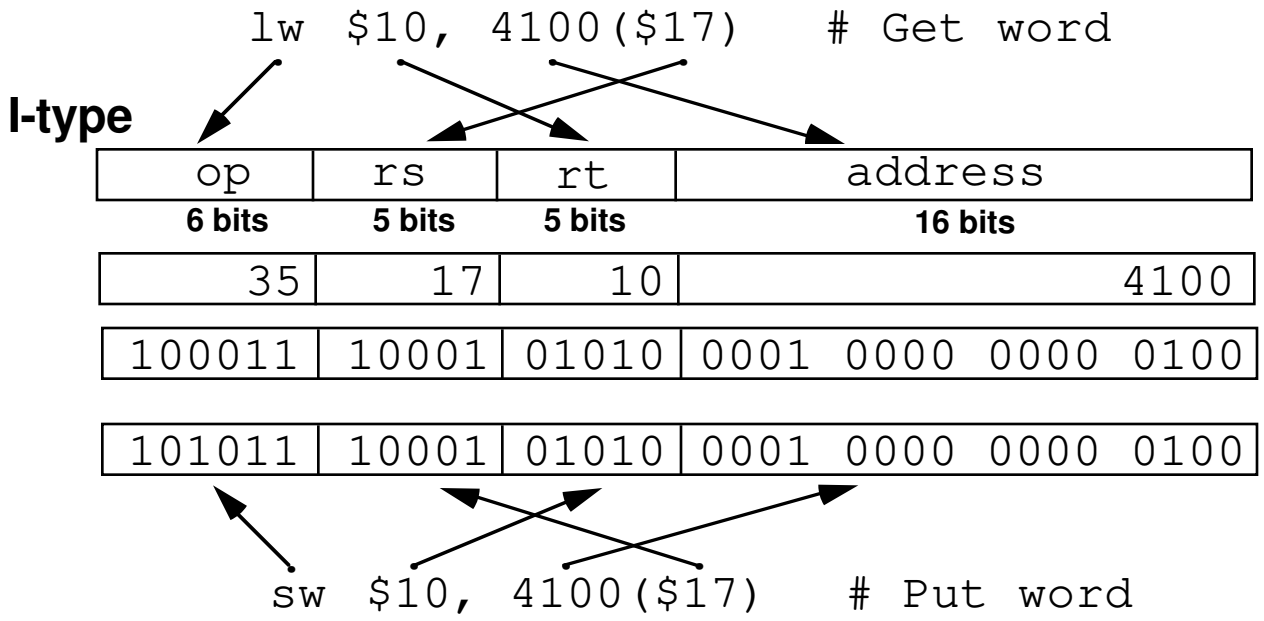
- Array indexing exploits base addressing: keep the index in register
  - Needs 0 origin indexing
  - Convert item number to byte offset

```
muli $19, $18, 4 # S'pose i in $18; find offset
lw  $8,8200($19)  # S'pose A starts at 8200; find A[i]
```


© 1995 LS All Rights Reserved

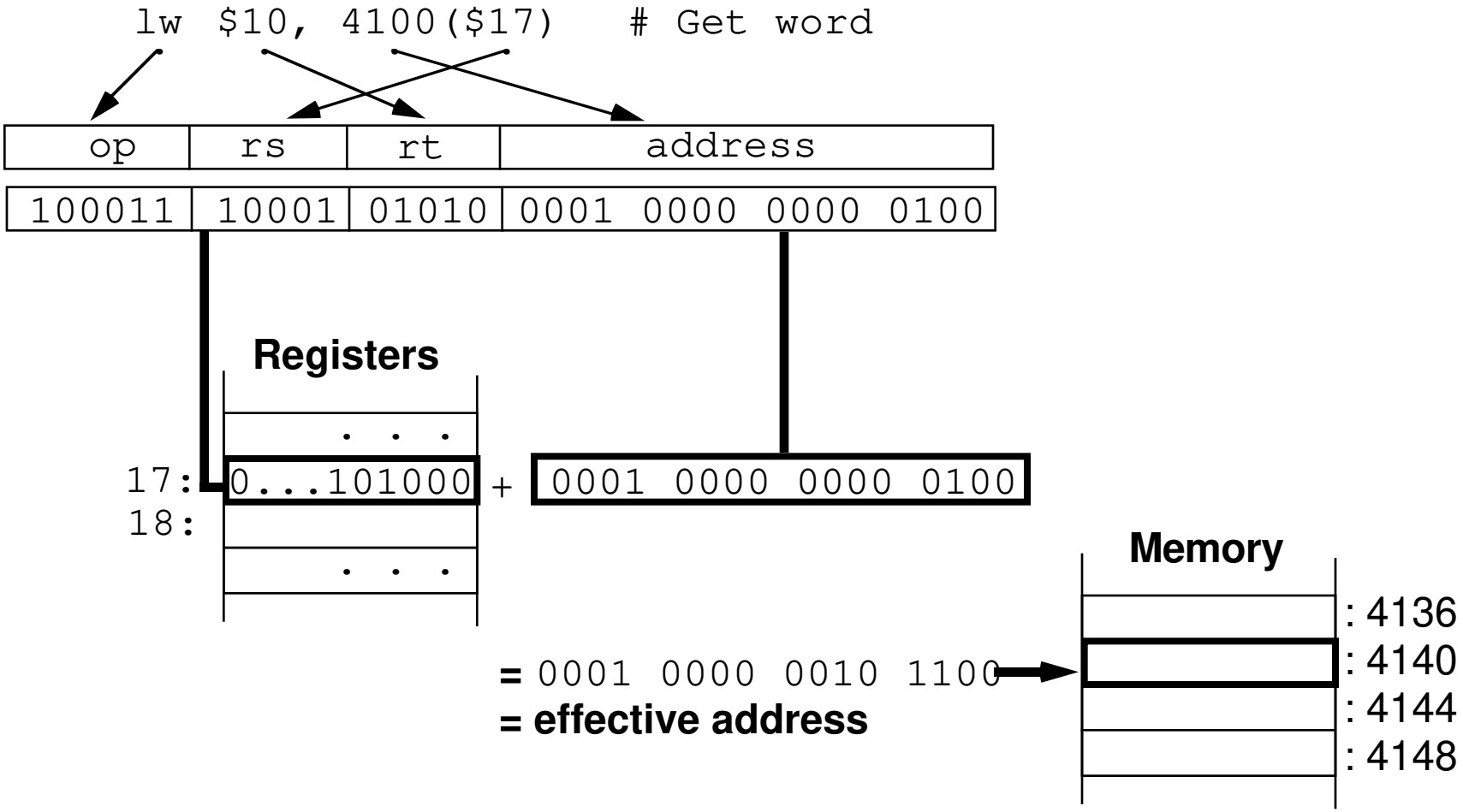
Load/Store

Load/Store instructions are I-type formats



© 1995 LS All Rights Reserved

Base Addressing



© 1995 LS All Rights Reserved

Control Flow

Branch instructions test to see if a register equals 0 (I-type)

```
        beq $8, $9, Label    # goto Label if $8=$9

        bne $5, $4, Label    # goto Label if $4≠$5

    Label: . . .
```

Comparisons between registers uses "set less than" (R-type)

```
        slt $1, $4, $5       # $1 gets 1 if $4<$5

        bne $1, $0, LT       # branch if condition true
```

The assembler implements "branch on less than," `blt`, as above
Register $1 reserved for the assembler's use in such cases
There is also an `slti`, for comparison with small constants

```
        slti   $1, $10, 144 # set $1 if $10 not gross
```


© 1995 LS All Rights Reserved

Example -- Sum some numbers

Problem: Add numbers in a 20 element array A, and save them in Total

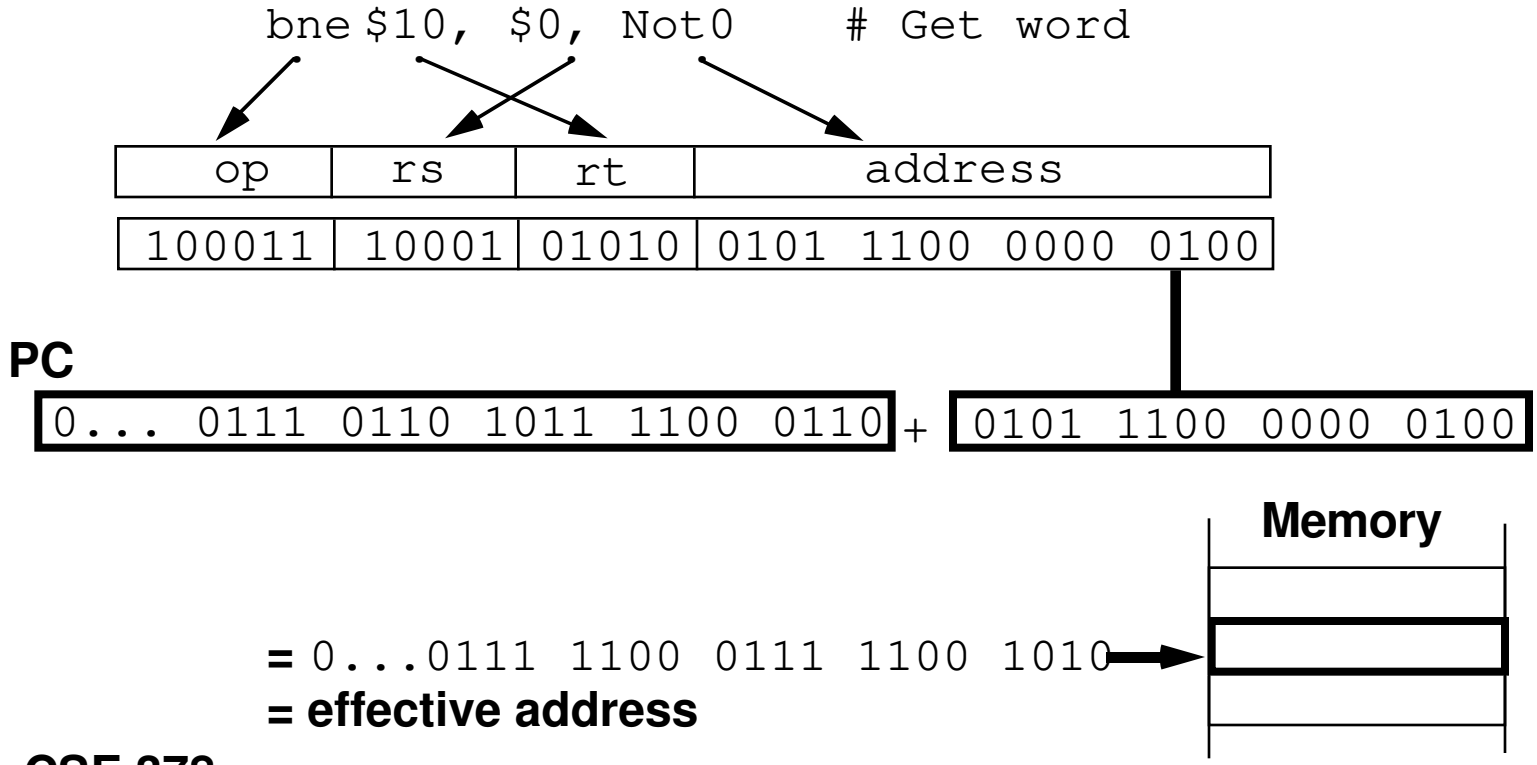
Solution: Initialize, load item, add in, inc index by 4, test, loop or save

```
      add    $10, $0, $0   #initialize running sum
      add    $11, $0, $0   #intialize index
L:    lw     $12,  A($11)  #get array item
      add    $10, $10, $12 #accumulate value
      addi   $11, $11, 4   #bump index to next item
      slti   $13, $11, 80  #set if items left (20*4)
      bne    $13, L        #iterate if not done
      sw     $12, $0, Total #save result
```


© 1995 LS All Rights Reserved

Branch Address Computation

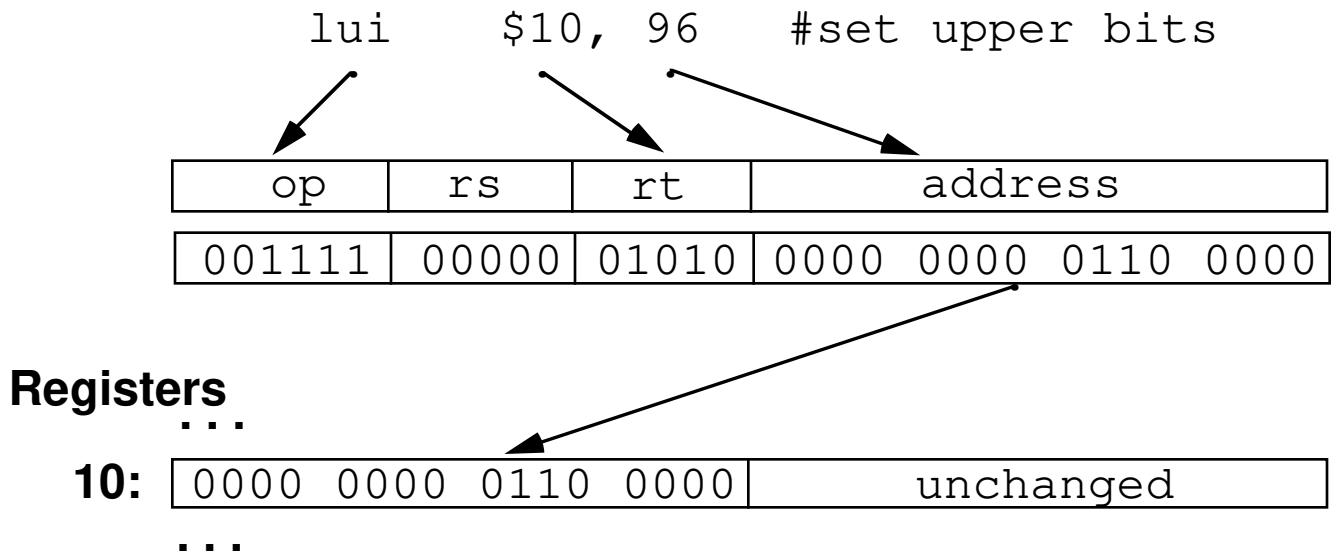
How many instructions can be addressed with 16 bit address field?
How far do branches typically have to go?
Branches (`beq` and `bne`) us an addressing mode called PC-relative
PC := PC + 4 + branch address field



© 1995 LS All Rights Reserved

Lui, Lui

The limitations of a 16-bit immediate field are eased by Load Upper Immediate (I-type).



Full word constants can be created using `addi` and `lui`.

© 1995 LS All Rights Reserved