Lesson 3

Signed and Unsigned Numbers

Representation of integers base 2, 10, and 16, conversion between bases. Binary addition and subtraction

Day 2 Review - Assembly Language

- To command a computer’s hardware, you must speak its language.
- The words of a computer’s language are called instructions, and its vocabulary is called an instruction set.
- Computer instructions can be represented as sequences of bits. This representation is called machine language
- Stored-program computer is the idea that instructions and data of many types can be stored in memory as numbers

Day 2 Review - Operations Of The Computer Hardware

- Every computer must be able to perform arithmetic
- Example of MIPS assembly language notation:

  add a, b, c

  The above means add b and c and put the result in a
- In MIPS, data must be in registers to perform arithmetic

Day 2 Review - The MIPS Assembly Language Notation

- add a, b, c
- The above MIPS assembly language notation instructs a computer to add the two variables b and c and to put their sum in a.
- Type a MIPS add instruction that computes:
  z = x + y
- Solution: add z, x, y or add z y, x
- The natural number of operands for an operation like addition is three: the two numbers being added together and a place to put the sum.

Day 2 Review - Example 1 of compiling a complex C assignment into MIPS.

- C Code
  - f = (g + h) - (i + j);
- MIPS
  - add t0, g, h # temp t0 contains g + h
  - add t1, i, j # temp t1 contains i + j
  - sub f, t0, t1 # f gets t0 - t1, which is (g + h) - (i + j)

Day 2 Review - Constant or immediate operands

- Many times a program will use a constant in an operation
- *add immediate* or addi.
- addi $s3, $s3, 4 # $s3 = $s3 + 4

Day 2 Review - Examples

- add $s1, $s2, $s3 # add
- addi $s1, $s3, 50 # add immediate

Day 2 **Review** - Load word instruction.

g = h + A[8];

$s1	60	g
$s2	35	h
$s3	5000	A's base addr
$t0	25	

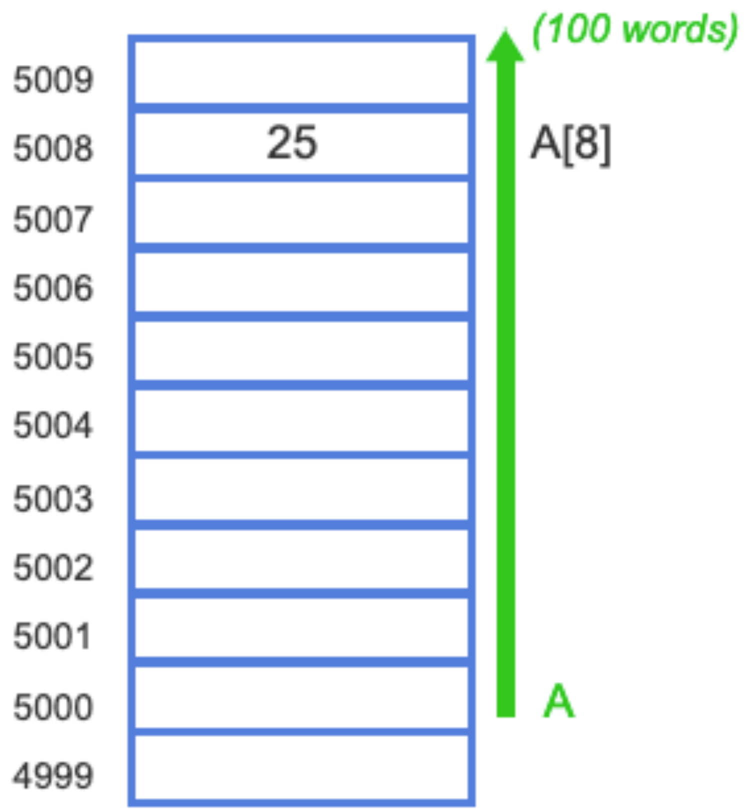
Registers

```
# Temporary reg $t0 gets A[8]
lw     $t0, 8($s3)
```

$8 + 5000$

```
# g = h + A[8]
add    $s1, $s2, $t0
```

35 + 25



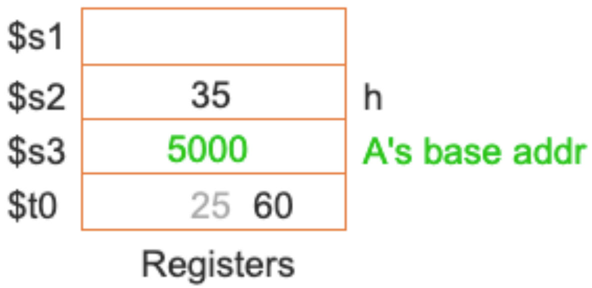
Memory

Day 2 Review - Store word

- Load word and store word are the instructions that copy words between memory and registers in the MIPS architecture.
- Store copies data from a register to memory
- The actual MIPS name is sw, standing for *store word*.

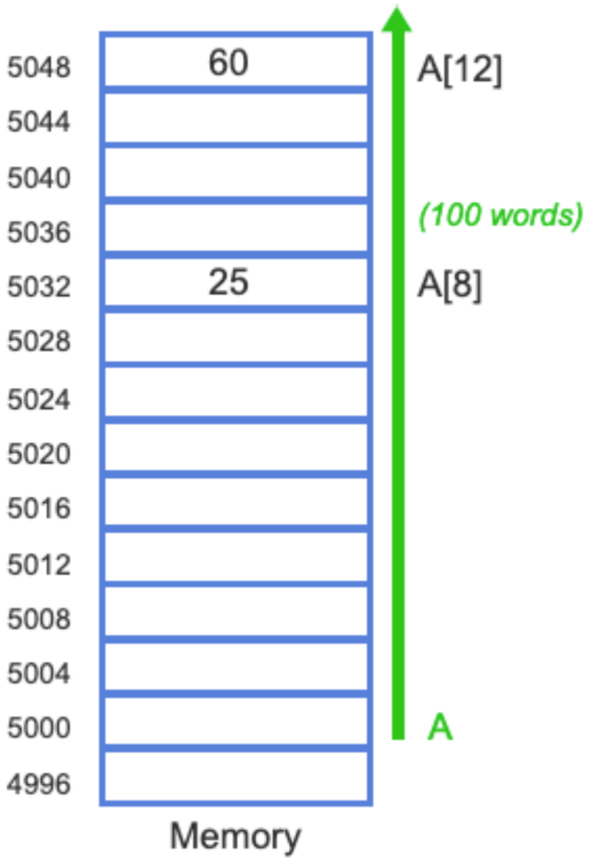
Day 2 Review - Example of compiling using load and store.

A[12] = h + A[8];




```
lw    $t0, 32($s3)   # Temporary reg $t0 gets A[8]
add   $t0, $s2, $t0  # Temporary reg $t0 gets h + A[8]

sw    $t0, 48($s3)   # Stores h + A[8] back into A[12]
```



Signed and unsigned numbers

- Let's quickly review how a computer represents numbers. Humans are taught to think in base 10, but numbers may be represented in any base. For example, 123 base 10 = 1111011 base 2.
- Numbers are kept in computer hardware as a series of high and low electronic signals, and so they are considered base 2 numbers.
- A single digit of a binary number is thus the "atom" of computing, since all information is composed of binary digits or bits.
- ***Binary digit***: Also called a bit. One of the two numbers in base 2 (0 or 1) that are the components of information.
- Just as base 10 numbers are called decimal numbers, base 2 numbers are called binary numbers.

Binary Numbers(Review)

- In assembly language it is important to remember that the actual hardware to be used only understands binary values 0 and 1.
- Binary values 0 and 1 are called binary numbers
- The numbering system that everyone learns in school is called decimal or base 10.
- The numbering system is called decimal because it has 10 digits, [0..9].
- Binary Numbers uses base 2 numbering system
- In binary, there are only two digits, 0 and 1.

How does a computer represent numbers? (Review)

- Numbers are kept in computer hardware as a series of high and low electronic signals, and so they are considered base 2 numbers
- Computers use switches that can be either on (1) or off(0), and so computers use the binary numbering system (base 2 numbering system)
- A single digit of a binary number is thus the "atom" of computing, since all information is composed of binary digits or bits.
- ***Binary digit***: Also called a bit. One of the two numbers in base 2 (0 or 1) that are the components of information.

How to convert binary to decimal

- decimal = $d_0 \times 2^0 + d_1 \times 2^1 + d_2 \times 2^2 + \ldots$
- Generalizing the point, in any number base, the value of *i* th digit *d* is
  $d \times Base^i$

Example:

$$1011_{two}$$

$$(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)_{ten}$$

$$= (1 \times 8) + (0 \times 4) + (1 \times 2) + (1 \times 1)_{ten}$$

$$= 8 + 0 + 2 + 1_{ten}$$

$$= 11_{ten}$$

Example 1:

- What is 1110_{two} in base ten?
- Solution:

$(1 * 2^3) + (1 * 2^2) + (1 * 2^1) + (0 * 2^0)$

$(1 * 8) + (1 * 4) + (1 * 2) + (0 * 1)$

$8 + 4 * 2 + 0$

14

Example 2:

- What is 1011_{two} in base ten?
- Solution:

  $(1 * 2^3) + (0 * 2^2) + (1 * 2^1) + (1 * 2^0)$

  $(1 * 8) + (0 * 4) + (1 * 2) + (1 * 1)$

  $8 + 0 * 2 + 1$

  11
- What is 111001_{two} in base ten?
- What is 100011_{two} in base ten?

Example 3:

- What is 111001_{two} in base ten?
- Solution:

  $(1 * 2^5) + (1 * 2^4)$ $+ (1 * 2^3)$ $(0 * 2^2) + (0 * 2^1)$ $+ (1 * 2^0)$

  $(1 * 32)$ $+ (1 * 16) + (1 * 8) + (0 * 4) + (0 * 2) + (1 * 1)$

  $32 + 16 * 8 + 0 + 0 + 1$

  57
- What is 100011_{two} in base ten?

Example 4:

- What is 100011_{two} in base ten?
- Solution:

  $(1 * 2^5) + (0 * 2^4)$ $+ (0 * 2^3)$ $(0 * 2^2) + (1 * 2^1)$ $+ (1 * 2^0)$

  $(1 * 32)$ $+ (0 * 16) + (0 * 8) + (0 * 4) + (0 * 2) + (1 * 1)$

  $32 + 0 * 0 + 0 + 2 + 1$

  25

Decimal to Binary Conversions

- Conversion steps:
    1. Divide the number by 2.
    2. Get the integer quotient for the next iteration.
    3. Get the remainder for the binary digit.
    4. Repeat the steps until the quotient is equal to 0.

Example 1

- Convert 13_{10} to binary:

Division by 2	**Quotient**	**Remainders**	**Bit #**
13/2	6	1	0
6/2	3	0	1
3/2	1	1	2
1/2	0	1	3

Writing the remainders from bottom to top, we have: $13_{10} = 1101_2$

Example 2

- Convert 147_{10} to binary:

Division by 2	**Quotient**	**Remainders**	**Bit #**
147/2	73	1	0
73/2	36	1	1
36/2	18	0	2
18/2	9	0	3
9/2	4	1	4
4/2	2	0	5
2/2	1	0	8
1/2	0	1	7

Writing and reading the remainders from bottom to top, we have: $147_{10} = 10010011_2$

Converting Fractions

- Fractions in any base system can be approximated in any other base system using negative powers of a radix.
- Radix points separate the integer part of a number from its fractional part.
- In the decimal system, the radix point is called a decimal point. Binary fractions have a binary point

Conversion steps for fractions:

1. Multiply the fractional decimal number by 2.
2. Integral part of resultant decimal number will be first digit of fraction binary number.
3. Repeat step 1 using only fractional part of decimal number and then step 2.

Example 1

- Convert 0.34375_{10} to binary with 4 bits to the right of the binary point.

$$\begin{array}{r l}
.34375 & \\
\underline{\times \quad\quad 2} & \\
0.68750 & \text{(Another placeholder.)} \\
.68750 & \\
\underline{\times \quad\quad 2} & \\
1.37500 & \\
.37500 & \\
\underline{\times \quad\quad 2} & \\
0.75000 & \\
.75000 & \\
\underline{\times \quad\quad 2} & \\
1.50000 & \text{(This is our fourth bit. We will stop here.)}
\end{array}$$

Reading from top to bottom, 0.3437510 = 0.0101_2 to four binary places.

Hexadecimals

- A Hexadecimal Number is based on the number **16**
- There are **16** Hexadecimal digits. They are the same as the decimal digits up to 9, but then there are the letters A, B, C, D, E and F in place of the decimal numbers 10 to 15:

Hexadecimal:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	14

Some Numbers to remember

Decimal	4-Bit Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Powers of 2
$2^{-2} = \frac{1}{4} = 0.25$
$2^{-1} = \frac{1}{2} = 0.5$
$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1{,}024$
$2^{15} = 32{,}768$
$2^{16} = 65{,}536$

Convert Binary to Hexadecimal

- Binary numbers are often expressed in hexadecimal to improve their readability.
- Example: Convert 110010011101_2 to hexadecimal.
- Solutions:

$$\underline{1100}\ \underline{1001}\ \underline{1101}$$
$$\quad C \quad\ \ 9 \quad\ \ D$$

Separate into groups of 4 for the hexadecimal conversion.

$$110010011101_2 = C9D_{16}$$

- If there are too few bits, leading zeros can be added.

Signed and Unsigned Numbers

- Computer programs calculate both positive and negative numbers, so we need a representation that distinguishes the positive from the negative
- Unsigned numbers stored only positive numbers but do not store negative numbers.
- Signed numbers use sign flag or can be distinguish between negative values and positive values.
- Number representation techniques like: Binary, Decimal and Hexadecimal number representation techniques that we have discussed above can represent numbers in both signed and unsigned ways

signed binary and Unsigned binary

- Unsigned binary numbers do not have sign bit
- Signed binary numbers uses signed bit and magnitude
  - The most obvious solution is to add a separate sign, which conveniently can be represented in a single bit; the name for this representation is sign and magnitude
  - Two’s complement representation: leading 0s mean positive, and leading 1s mean negative.

Unsigned Numbers:

- Unsigned numbers do not have any sign as they only represent positive numbers
- Represent decimal number 92_{10} in unsigned binary number.
  - $= (1x2^6+0x2^5+1x2^4+1x2^3+1x2^2+0x2^1+0x2^0)_{10}$
  - $= (1011100)_2$ a 7-bit binary magnitude of the decimal number 92_{10}.
- You can also used division to get to the above above answer

Signed Numbers:

- Signed numbers contain sign flag ro distinguish positive and negative numbers.
- This technique contains both sign bit and magnitude of a number.
- Three types of representations for signed binary numbers
    1. Sign-Magnitude form
    2. 1’s complement form
    3. 2’s complement form

Sign-Magnitude form

- For n bit binary number, 1 bit is reserved for sign symbol. If the value of sign bit is 0, the given number is positive, else if the value of sign bit is 1, the given number is negative.
- Remaining (n-1) bits represent magnitude of the number

Sign-Magnitude form Conversions Example

- What are the decimal values of the following 8-bit sign-magnitude numbers?
  - 10000011 = -3
  - 00000101 = +5
  - 11111111 = -127
  - 01111111 = 127

Sign-Magnitude form Conversions Example

- Represent the following in 8-bit sign-magnitude
  - -15 = 10001111
  - +7 = 00000111
  - -1 = 1001

1’s complement form

- Positive values in one's complement are the same as unsigned binary or sign-magnitude.
- To negate a one's complement value, we invert *all* bits. Like sign-magnitude, one's complement has two representations for zero (all zeros or all ones).

Conversions Example

- What are the decimal values of the following 8-bit one's complement numbers?
  - 00001010 = +10
  - 10001010 = -(01110101) = -(1+4+16+32+64) = -117
  - 11111111 = ?

2's Complement Form

- Commonly used
- A positive integer in two's complement always has a 0 in the leftmost bit (*sign bit*) and is represented the same way as an unsigned binary integer.
- To negate a number, a process sometimes called “taking the two's complement”, we invert all the bits and add one.

Conversion Example

- Example 1
  - $+14_{10} = 01110_{\text{two's comp}}$

- Example 2:
  - $-14_{10} = 10001 + 1$
  - $= 10010_{\text{two's comp}}$

- Convert the following 4-bit 2's comp values to decimal:
  - 0111 = +(1 + 2 + 4) = +7
  - 1000 = -(0111 + 1) = -(1000) = -8
  - 0110 = +(2 + 4) = +6
  - 1001 = -(0110 + 1) = -0111 = -(1 + 2 + 4) = -7
  - 1110 = -(0001 + 1) = -0010 = -2

Readings

- Hennessy and Patterson Chapter 2.5 to 2.7