2010-05-06

2010年 前学期 TOKYO TECH

計算機アーキテクチャ 第一 (E)

4. データ形式

吉瀬 謙二 計算工学専攻
kise_at_cs.titech.ac.jp
W641講義室　木曜日13:20 ー 14:50

整数(integer)の表現

- コンピュータは決まったビット幅を単位としてデータを処理.
  - **例えば，8ビットコンピュータ は，8ビット単位で処理**
- nビットの整数表現は，2^n（2のn乗）種類の整数を表現できる.（しか表現できない！）
  - 8ビットであれば，2^8 ＝ 256 種類の整数.
  - **表現できる範囲には限りがある.**
  - **効率の良い表現を利用して，資源を有効に活用する！**
- 整数表現
  - **符号なし表現**
  - **符号つき絶対値表現**
  - **2の補数表現**

整数：2の補数表現（1）

- 多くの計算機では2の補数 (two's complement) 表現が利用される.
- 2の補数の利点
  - 最上位ビットのみで正負判定が可能.
  - 正負の反転が容易.
  - ビット幅の異なるデータへの変換が容易.
  - 符号なし整数と同じハードウェアで加算を実装できる.

整数：2の補数表現（2）

- その前に，1の補数 (one's complement)
  - 全てのビットを反転することで，マイナスを表現

128種類

$0000\ 0000_2 = +0_{10}$	$1111\ 1111_2 = -0_{10}$
$0000\ 0001_2 = +1_{10}$	$1111\ 1110_2 = -1_{10}$
$0000\ 0010_2 = +2_{10}$	$1111\ 1101_2 = -2_{10}$
…	…
$0111\ 1101_2 = +125_{10}$	$1000\ 0010_2 = -125_{10}$
$0111\ 1110_2 = +126_{10}$	$1000\ 0001_2 = -126_{10}$
$0111\ 1111_2 = +127_{10}$	$1000\ 0000_2 = -127_{10}$

整数：2の補数表現（3）

- 2の補数
  - （1の補数で表された数に1を加えたもの）を負の数とする.

$0000\ 0000_2 = +0_{10}$	$1111\ 1111_2 = -0_{10}$	
$0000\ 0001_2 = +1_{10}$	$1111\ 1110_2 = -1_{10}$	$1111\ 1111_2 = -1_{10}$
$0000\ 0010_2 = +2_{10}$	$1111\ 1101_2 = -2_{10}$	$1111\ 1110_2 = -2_{10}$
…	…	…
$0111\ 1101_2 = +125_{10}$	$1000\ 0010_2 = -125_{10}$	$1000\ 0011_2 = -125_{10}$
$0111\ 1110_2 = +126_{10}$	$1000\ 0001_2 = -126_{10}$	$1000\ 0010_2 = -126_{10}$
$0111\ 1111_2 = +127_{10}$	$1000\ 0000_2 = -127_{10}$	$1000\ 0001_2 = -127_{10}$
	負の数の1の補数表現	$1000\ 0000_2 = -128_{10}$
		負の数の2の補数表現

2の補数では，ー128 ～ 127 までの数を表現できる.

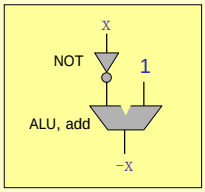
整数：2の補数表現（4）

- 2の補数
  - 1の補数で表された数（ビットの反転）に1を加えたものを負の数とする.

$0000\ 0000_2 = +0_{10}$	$1111\ 1111_2 = -0_{10}$	負の数の2の補数表現
$0000\ 0001_2 = +1_{10}$	$1111\ 1110_2 = -1_{10}$	$1111\ 1111_2 = -1_{10}$
$0000\ 0010_2 = +2_{10}$	$1111\ 1101_2 = -2_{10}$	$1111\ 1110_2 = -2_{10}$
…	…	…
$0111\ 1101_2 = +125_{10}$	$1000\ 0010_2 = -125_{10}$	$1000\ 0011_2 = -125_{10}$
$0111\ 1110_2 = +126_{10}$	$1000\ 0001_2 = -126_{10}$	$1000\ 0010_2 = -126_{10}$
$0111\ 1111_2 = +127_{10}$	$1000\ 0000_2 = -127_{10}$	$1000\ 0001_2 = -127_{10}$
		$1000\ 0000_2 = -128_{10}$

整数：2の補数表現（5）

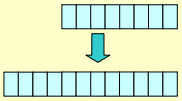
- 2の補数
  - 1の補数で表された数（ビットの反転）に1を加えたものを負の数とする．
- 2の補数表現では，正負の反転を簡潔に実現できる！
  - 正数から負数への変換
    - 2進数表現の1と0を反転する．
    - 得られたデータに1を加える．
  - 負数から正数への変換
    - 2進数表現の1と0を反転する．
    - 得られたデータに1を加える．



7

整数：2の補数表現（6）

- 符号拡張
  - ビット幅の異なるデータへの変換
  - 例：8ビットから12ビットのデータへの変換
- 符号拡張の処理
  - ビット幅を増やすときには，最上位ビットの値で補填すればよい．

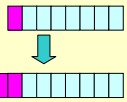


$1111\ 1111_2 = -1_{10}$		$1111\ 1111\ 1111_2 = -1_{10}$
$1111\ 1110_2 = -2_{10}$		$1111\ 1111\ 1110_2 = -2_{10}$
…	符号拡張	…
$1000\ 0011_2 = -125_{10}$	⇒	$1111\ 1000\ 0011_2 = -125_{10}$
$1000\ 0010_2 = -126_{10}$		$1111\ 1000\ 0010_2 = -126_{10}$
$1000\ 0001_2 = -127_{10}$		$1111\ 1000\ 0001_2 = -127_{10}$
$1000\ 0000_2 = -128_{10}$		$1111\ 1000\ 0000_2 = -128_{10}$

8

整数：2の補数表現（7）

- 符号拡張
  - ビット幅の異なるデータへの変換
  - 例：8ビットから12ビットのデータへの変換
- 符号拡張の処理
  - ビット幅を増やすときには，最上位ビットの値で補填すればよい．



9

2の補数の加算（1）

- 符号を意識することなく，符号なし整数の加算と同様に計算できる．

桁上げ → 0 0 0 0　1 1 0

$$\begin{array}{rr} & 0000\ 0111_2 = 7_{10} \\ + & 0000\ 0110_2 = 6_{10} \\ \hline & 0000\ 1101_2 = 13_{10} \end{array}$$

10

2の補数の加算（2）

- 符号を意識することなく，符号なし整数の加算と同様に計算できる．

桁上げ → 1 1 1 1　1 1 0

$$\begin{array}{rr} & 0000\ 0111_2 = 7_{10} \\ + & 1111\ 1010_2 = -6_{10} \\ \hline & 0000\ 0001_2 = 1_{10} \end{array}$$

減算：X − Y = X + (− Y)

11

整数の表現のまとめ

- 符号なし表現
- 符号つき絶対値表現
- 1の補数表現
- 2の補数表現
  - 最上位ビットのみで正負判定が可能．
  - 正負の反転が容易．
  - ビット幅の異なるデータへの変換が容易．
  - 符号なし整数と同じハードウェアで符号付き加算を実装できる．

12

実数

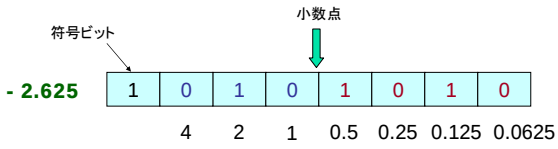
- 少数を含む数値を取り扱う.
- 実数の例
  - 3.1419926… (π)
  - 0.000000001, **1.0 x 10^{-9}**
  - 3,155,760,000, **3.1556 x 10^{9}**

科学記数法：小数点の左側には数字を一つしか書かない.
科学記数法で書いた数値で先頭に0がこないものを**正規化数**と呼ぶ.

13

固定小数点表現

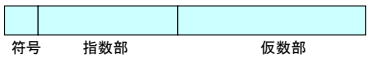
- あまり利用されない！
  - 小数点の位置を固定する.



14

浮動小数点表現(1)

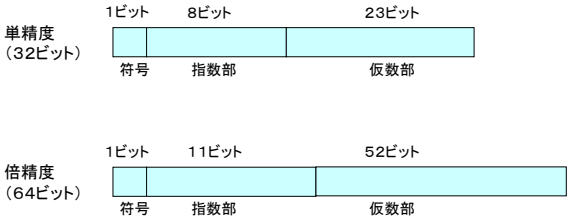
- 小数点位置が変動
- 科学記数法で数値で先頭に0がこない正規化数を利用.



15

浮動小数点表現(2)

- IEEE754



16

浮動小数点表現(3)

- 誤差
  - 実数は不可算無限
  - 決められたビットで表現できる数は有限
    - **丸め誤差**が発生
- 表現できないほど大きな数
- 表現できないほど小さな数
- 非常に大きな数と，非常に小さな数の間の演算

- 10進数で 0.10 は，
  2進数で 0.0001100110011… どうすれば良いか？

Packed decimal

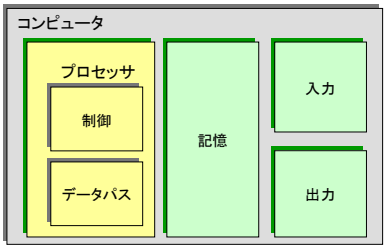
17

計算機アーキテクチャ 第一 (E)

4. プロセッサの動作原理

吉瀬 謙二 計算工学専攻
kise_at_cs.titech.ac.jp
W641講義室 木曜日13:20 ー 14:50

コンピュータ（ハードウェア）の古典的な要素



プロセッサは記憶装置から命令とデータを取り出す。入力装置はデータを記憶装置に書き込む。出力装置は記憶装置からデータを読みだす。制御装置は、データパス、記憶装置、入力装置、そして出力装置の動作を指定する信号を送る。

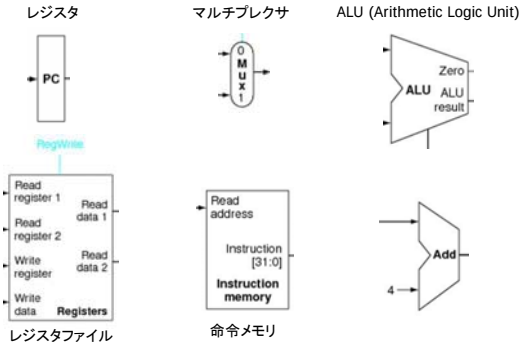
出典：パターソン & ヘネシー、コンピュータの構成と設計 19

MIPSの基本的な5つのステップ（ステージ）

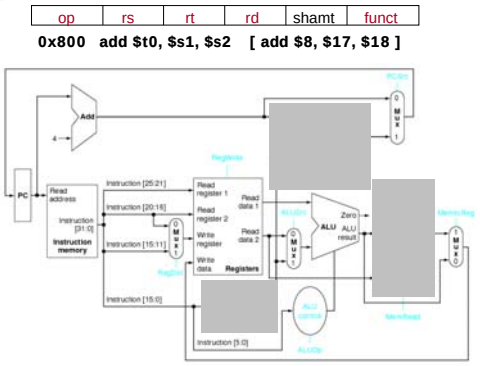
- **IFステージ**
  メモリから命令をフェッチする.
- **IDステージ**
  命令をデコード（解読）しながら，レジスタの値を読み出す.
- **EXステージ**
  命令操作の実行またはアドレスの生成を行う.
- **MEMステージ**
  必要であれば，データ・メモリ中のオペランドにアクセスする.
- **WBステージ**
  必要であれば，結果をレジスタに書き込む.

20

主な構成要素（1）

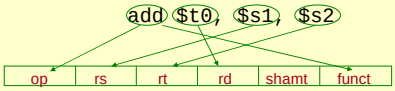


プロセッサのデータパス（シングル・サイクル）



Machine Language - Add Instruction

- Instructions, like registers and words of data, are **32 bits long**
- Arithmetic Instruction Format (R format):

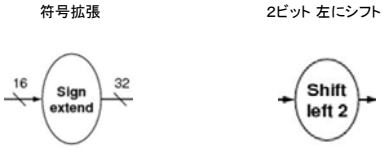


op	6-bits	opcode that specifies the operation
rs	5-bits	register file address of the first source operand
rt	5-bits	register file address of the second source operand
rd	5-bits	register file address of the result's destination
shamt	5-bits	shift amount (for shift instructions)
funct	6-bits	function code augmenting the opcode

23

Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

主な構成要素（2）



Exercise

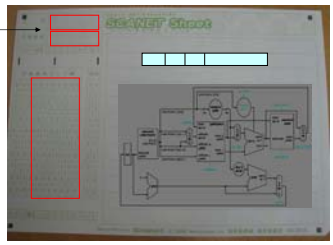
op	rs	rt	16 bit immediate

I format

0x804 addi $t0, $t1, -1 [addi $8, $9, -1]

氏名，学籍番号，
学籍番号マーク欄(**右詰で**)

32ビットの命令列を2進数で示せ．
この命令を実行する際，
確定するデータパスに値を示せ．

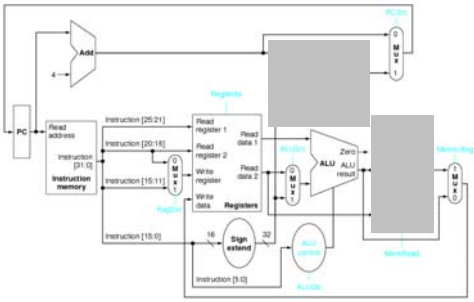


プロセッサの**データパス**(シングル・サイクル)

op	rs	rt	16 bit immediate

I format

0x804 addi $t0, $t1, -1 [addi $8, $9, -1]

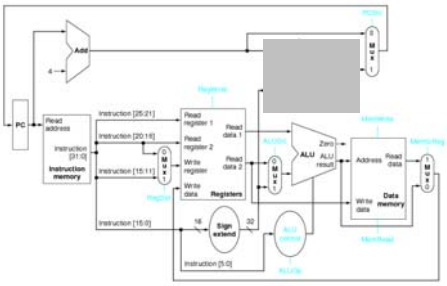


プロセッサの**データパス**(シングル・サイクル)

op	rs	rt	16 bit immediate

I format

0x808 lw $t0, 24($s2) [lw $8, 24($18)]



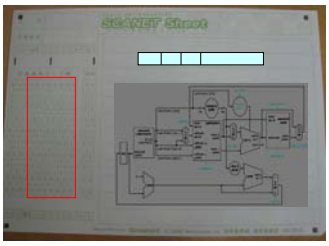
Exercise

op	rs	rt	16 bit immediate

I format

0x808 lw $t0, 24($s2) [lw $8, 24($18)]

32ビットの命令列を2進数で示せ．
この命令を実行する際，
確定するデータパスに値を示せ．

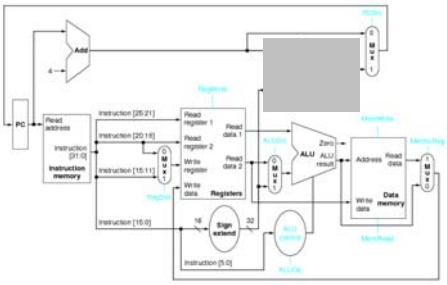


プロセッサの**データパス**(シングル・サイクル)

op	rs	rt	16 bit immediate

I format

sw $t0, 24($s2) [sw $8, 24($18)]

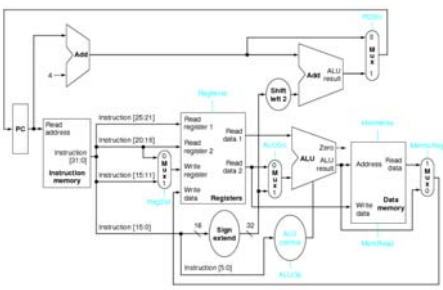


プロセッサの**データパス**(シングル・サイクル)

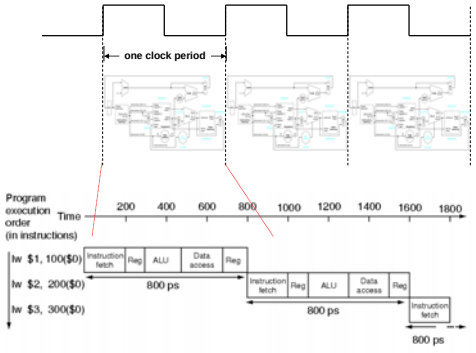
op	rs	rt	16 bit immediate

I format

beq $s0, $s1, Label [beq $16, $17, Label]



プロセッサのデータパス（シングル・サイクル）



アナウンス

- 講義スライドおよびスケジュール
  - www.arch.cs.titech.ac.jp
  - 講義日程が変更になることがあるので頻繁に確認すること.

32