2009-05-014

2009年 前学期 TOKYO TECH

計算機アーキテクチャ 第一 (E)

4. 命令形式, アドレス指定形式

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

Acknowledgement

- Lecture slides for Computer Organization and Design, Third Edition, courtesy of **Professor Mary Jane Irwin**, Penn State University
- Lecture slides for Computer Organization and Design, third edition, Chapters 1-9, courtesy of **Professor Tod Amon,** Southern Utah University.

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

参考書(読んでください)

- **コンピュータの構成と設計 第3版**、パターソン＆ヘネシー(成田光彰 訳)、日経BP社、2006
- コンピュータアーキテクチャ 定量的アプローチ 第4版 翔泳社, 2008
- コンピュータアーキテクチャ, 村岡 洋一 著, 近代科学社, 1989
- 計算機システム工学, 富田 眞治, 村上 和彰 著, 昭晃堂, 1988
- コンピュータハードウエア, 富田 眞治, 中島 浩 著, 昭晃堂, 1995
- 計算機アーキテクチャ, 橋本 昭洋 著, 昭晃堂, 1995

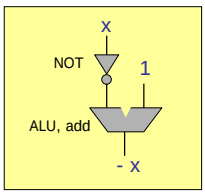
整数:2の補数表現(4)

- 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.

		負の数の2の補数表現
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$_{10}$
		1000 0000$_2$ = −128$_{10}$

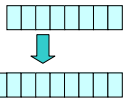
整数:2の補数表現(5)

- 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.



整数:2の補数表現(6)

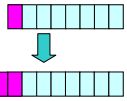
- 符号拡張
  - ビット幅の異なるデータへの変換
  - 例: 8ビットから12ビットのデータへの変換
- 符号拡張の処理
  - ビット幅を増やすときには, 最上位ビットの値で補填すればよい.



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$_{10}$

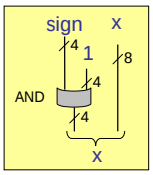
整数：2の補数表現（7）

- 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．



証明

ある数 X が正の数の場合には自明．
それから．．．



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

0 0 0 0　0 1 1 1 $_2$ = 7 $_{10}$

\+ 0 0 0 0　0 1 1 0 $_2$ = 6 $_{10}$

0 0 0 0　1 1 0 1 $_2$ = 13 $_{10}$

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

0 0 0 0　0 1 1 1 $_2$ = 7 $_{10}$

\+ 1 1 1 1　1 0 1 0 $_2$ = -6 $_{10}$

0 0 0 0　0 0 0 1 $_2$ = 1 $_{10}$

減算：X – Y = X + (- Y)

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．

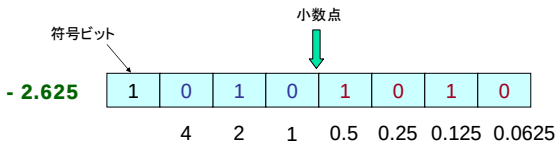
実数

- 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．

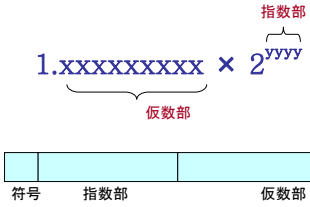
固定小数点表現

- あまり利用されない！
  - 小数点の位置を固定する．



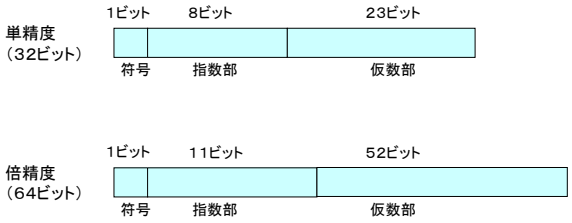
浮動小数点表現(1)

- 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.



浮動小数点表現(2)

- IEEE754



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
    - 対応がうまくいかない多くで, **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

講義用の計算機環境

- 講義用の計算機
  - 131.112.16.56
  - ssh arche@131.112.16.56
    - **ユーザ名: arche**
    - **パスワードは講義時に連絡**
  - mkdir myname (例: mkidr 06B77777)
  - cd myname (例: cd 06B77777)
- 注意点
  - 計算機演習室からは外部にsshで接続できないかもしれません.
  - Windowsからは Tera Term などを利用してください.

Sample program

コンパイラの最適化オプションを変更しながら, どのような命令列が出力されるか試してみる.

```
#include <stdio.h>
int main(){
  int i;
  int sum = 0;

  for(i=1; i<=100; i++)  sum += i;

  return sum;
}
```

mipsel-linux-gcc -O0 -S main.c -o main_opt0.s
/home/share/cad/mipsel/usr/bin/mipsel-linux-gcc

レポート 問題

1. `void max (int v[], int n)`
   をクロスコンパイラにてMIPS命令セットにコンパイルし, コンパイルオプションによってどのように変化するかをまとめよ.
2. `void sort (int v[], int n)`
   をクロスコンパイラにてMIPS命令セットにコンパイルし, コンパイルオプションによってどのように変化するかをまとめよ.
3. 同様に, サンプルアプリケーションを作成し, それをクロスコンパイラにてMIPS命令セットにコンパイルし, コンパイルオプションによってどのように変化するかをまとめよ.
4. この課題の感想をまとめること.
5. レポートはA4用紙2枚以内にまとめること. (必ずPDFとすること)
   (2段組, コードは小さい文字でもかまわない.)

レポート 提出方法

- 5月13日（午後7時）までに電子メールで提出
  - 人よりも先に提出している（先願性）と高得点
  - report_at_arch.cs.titech.ac.jp
- 電子メールのタイトル
  - Arch Report [学籍番号]
  - 例：Arch Report [33_77777]
- 電子メールの内容
  - 氏名，学籍番号
  - 回答
    - PDFファイルを添付（必ずPDFとすること）
    - PDFファイルにも氏名，学籍番号を記入すること．
    - A4用紙で2枚以内にまとめること．

2009-05-014

計算機アーキテクチャ 第一 (E)

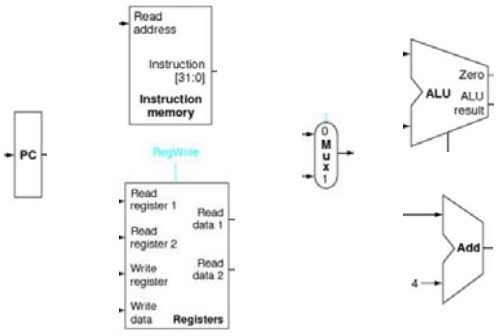
プロセッサの原理

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

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**
  命令をデコードしながら，レジスタを読み出す．
- **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**
  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**
  結果をレジスタに書き込む．

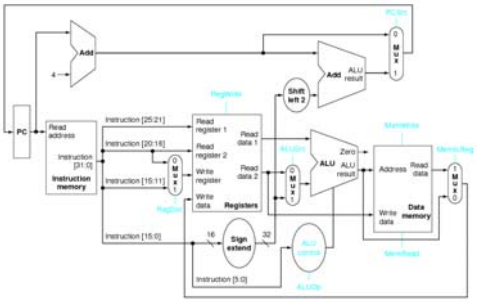
プロセッサの主な構成要素



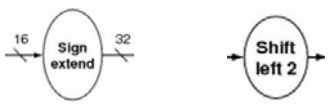
プロセッサの**データパス**（シングル・サイクル）

op	rs	rt	rd	shamt	funct

add $t0, $s1, $s2　[add $8, $17, $18]



プロセッサの構成要素（1）



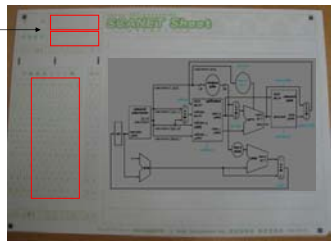
Exercise

op	rs	rt	16 bit immediate

I format

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**)

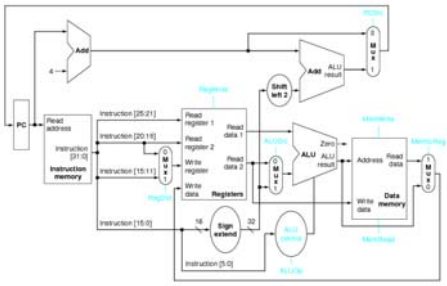


プロセッサの**データパス**(シングル・サイクル)

op	rs	rt	16 bit immediate

I format

addi $sp, $sp, 4 **[addi $29, $29, 4]**

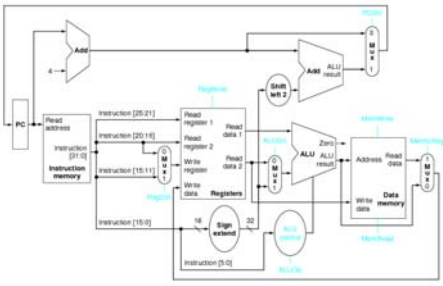


プロセッサの**データパス**(シングル・サイクル)

op	rs	rt	16 bit immediate

I format

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

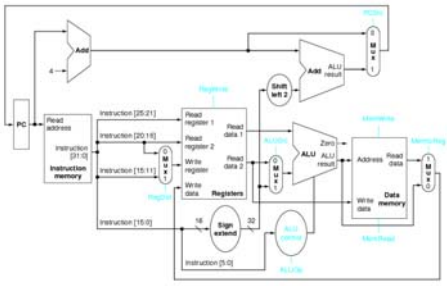


プロセッサの**データパス**(シングル・サイクル)

op	rs	rt	16 bit immediate

I format

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

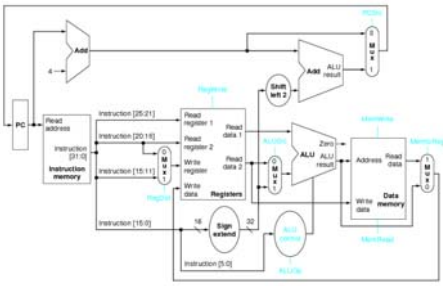


プロセッサの**データパス**(シングル・サイクル)

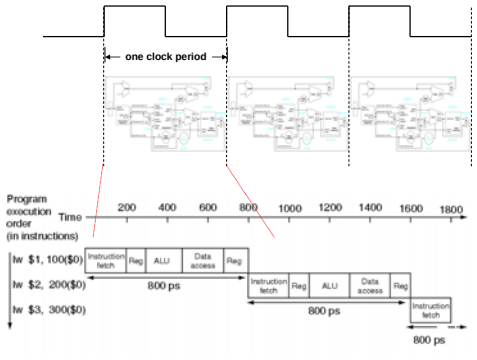
op	rs	rt	16 bit immediate

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)



Sample program

```
#include <stdio.h>
int main(){
  int i;
  int sum = 0;

  for(i=1; i<=100; i++)  sum += i;

  return sum;
}
```

コンパイラの最適化オプションを変更しながら，SimMipsで実行し，その実行サイクル数をみる．

mipsel-linux-gcc –static –O0 main.c –o a.out
SimMips a.out
/home/share/cad/mipsel/usr/bin/mipsel-linux-gcc

31

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

レポート 問題

1. void max (int v[], int n)
   をクロスコンパイラにてMIPS命令セットにコンパイルし，コンパイルオプションによってどのように変化するかをまとめよ．また，SimMipsで実行し，実行サイクル数を比較せよ．
2. void sort (int v[], int n)
   をクロスコンパイラにてMIPS命令セットにコンパイルし，コンパイルオプションによってどのように変化するかをまとめよ．また，SimMipsで実行し，実行サイクル数を比較せよ．
3. 同様に，**複雑なアプリケーション**を作成し，それをクロスコンパイラにてMIPS命令セットにコンパイルし，コンパイルオプションによってどのように変化するかをまとめよ．また，SimMipsで実行し，実行サイクル数を比較せよ．
4. この課題の感想をまとめること．
5. レポートはA4用紙3枚以内にまとめること．（必ずPDFとすること）
   （2段組，コードは小さい文字でもかまわない．）

レポート 提出方法

- 5月21日（午後7時）までに電子メールで提出
  - 人よりも先に提出している（先願性）と高得点
  - report_at_arch.cs.titech.ac.jp
- 電子メールのタイトル
  - Arch Report [学籍番号]
  - 例：Arch Report [33_77777]
- 電子メールの内容
  - 氏名，学籍番号
  - 回答
    - PDFファイルを添付（必ずPDFとすること）
    - PDFファイルにも氏名，学籍番号を記入すること．
    - A4用紙で3枚以内にまとめること．