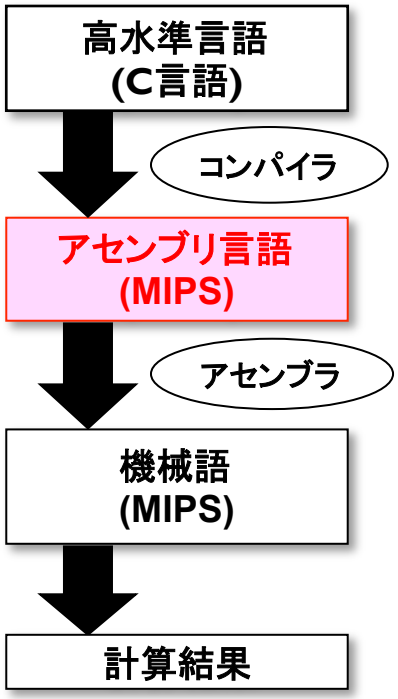
2012年度
計算機システム演習　第4回
2012.05.07

白幡　晃一

第２回課題の補足

- TSUBAMEへのログイン
  - TSUBAMEは学内からのログインはパスワードで可能
    - しかし、演習室ではパスワードでログインできない設定
    - 公開鍵認証でログイン
  - 公開鍵，秘密鍵の生成
    - ターミナルを開く
    - $ ssh-keygen
    - Enter file in which to save the key (/Users/shirahata/.ssh/id_rsa):
      - Return
    - Enter passphrase (empty for no passphrase):
      - 公開鍵のパスワードを入力
    - Enter same passphrase again:
      - もう一度同じパスワードを入力
    - 「id_rsa」が秘密鍵、「id_rsa.pub」が公開鍵になる。
  - TSUBAMEへの公開鍵の登録
    - 東工大ポータル -> TSUBAME2.0 利用ポータル -> SSH公開鍵アップロード
    - 公開鍵を選択しアップロード(生成した公開鍵 id_rsa.pub をアップロード)
  - 以上の操作により演習室からTSUBAMEへログインできるようになる



今日の内容

アセンブラ命令 2
配列の操作方法

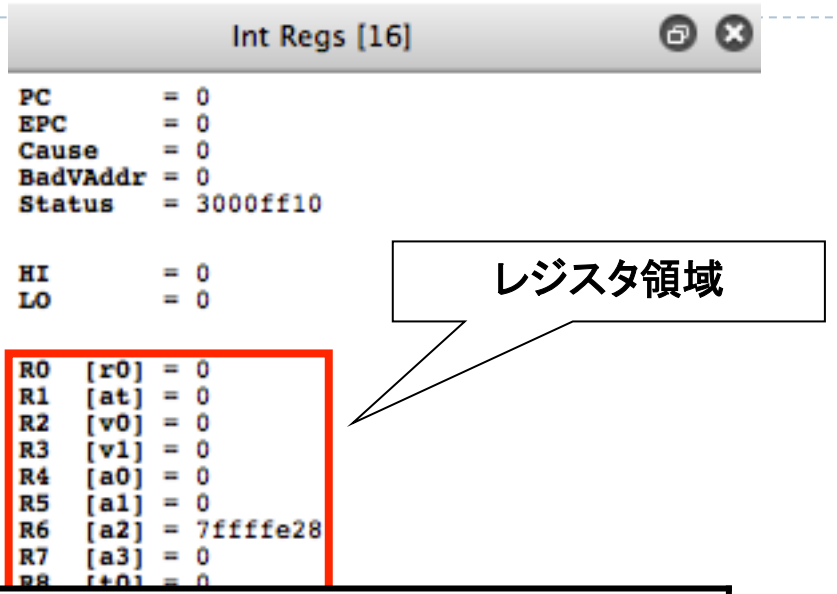
加減算

- add $A, $B, $C
  - $A <- $B + $C
- addi $A, $B, 数値
  - 即値可算(add immediate)
    - レジスタが示す値に定数を加算
  - $A <- $B + 数値
- sub $A, $B, $C
  - $A <- $B - $C
- subiは無い
  - addiで数値に負の値を指定

```
add $t0, $t1, $t2
addi $t0, $t1, 4
sub $t0, $t1, $t2
addi $t0, $t1, -16
```

レジスタ

- CPU内部の記憶素子
  - MIPSでは32本



$a0 ~ $a3	サブルーチンの引数を入れる
$v0 ~ $v1	サブルーチンの返り値が入る
$t0 ~ $t9	一時変数（自由に使える）
$ra	（メイン）ルーチンから戻るアドレスを保持
$zero	定数値0

よく使う命令

- 分岐命令 (ジャンプ命令)
  - j, jr
- 条件分岐命令
  - beq, bne, blt, ble, bgt, bge
- 比較命令
  - slt, slti

分岐命令(1/2)

- `j label`
  - ラベルの命令へジャンプ

```
        j  next
           :
next:
           :
```

m3.s

```
            .data
J:
            .asciiz "Jump\n"
NJ:
            .asciiz "Not Jump\n"

            .text
main:
            j jump

            li $v0, 4
            la $a0, NJ
            syscall
            jr $ra

jump:
            li $v0, 4
            la $a0, J
            syscall
            jr $ra
```

※ labelの有無に関係なく、jump命令が呼び出されるまで、次の命令が実行され続ける

Jump

分岐命令(2/2)

- **`jr $A`**
  - レジスタ $A の値の指すアドレスにジャンプ
  - 例：`jr $ra`

```
        la   $t0, next
        jr   $t0
           :
next:
```

m4.s

```
            .data
J:
            .asciiz "Jump\n"
NJ:
            .asciiz "Not Jump\n"

            .text
main:
            la $t0, jump
            jr $t0

            li $v0, 4
            la $a0, NJ
            syscall
            jr $ra

jump:
            li $v0, 4
            la $a0, J
            syscall
            jr $ra
```

```
Jump
```

条件分岐命令

- **`beq $A, $B, label`**
  - branch on equal
  - $A == $B ならラベルにジャンプ
- **`bne $A, $B, label`**
  - branch on not equal
  - $A != $B ならラベルにジャンプ

```
      beq  $t0, $t1, Label
           :
Label:
           :
```

if 文の実現

```
if (x != 0)
    y = 1;
else
    y = 2;
```

```
    bne   $t0, $zero, then   # if (x!=0) goto then
    li    $t1, 2             # y=2
    j     end
then:
    li    $t1, 1             # y=1
end:
```

（$t0にx, $t1にyが該当する）

while 文の実現

```
while (x != y) {
    y++;
}
```



```
while (true) {
    if (x==y) break;
    y++;
}
```



```
while:
    beq  $t0, $t1, end   # if (x==y) goto end
    addi $t1, $t1, 1     # y++
    j    while
end:
```

（$t0にx, $t1にyが該当する）

比較命令

- **`slt $A, $B, $C`**
  - set less than
  - $B < $C なら $A = 1; そうでなければ $A = 0
- **`slti $A, $B, 数値`**
  - set less than immediate
  - $B < 数値 なら $A = 1; そうでなければ $A = 0

その他の条件分岐命令

`blt $A, $B, label`	$A < $B なら分岐
`ble $A, $B, label`	$A <= $B なら分岐
`bgt $A, $B, label`	$A > $B なら分岐
`bge $A, $B, label`	$A >= $B なら分岐

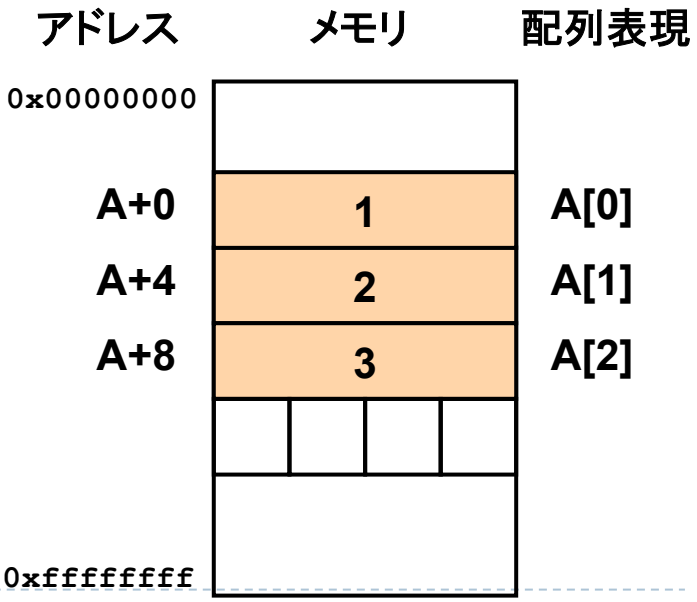
(less than, less than equal, greater than, greater than equal)

- これらは疑似命令
  - slt, beq, bne の組み合わせで実現できる (⇒課題)
  - 「move $A, $B」も擬似命令
    - add A, B, $zero
- その他の命令
  - http://www.cs.wisc.edu/~larus/HP_AppA.pdf の A.10 (A-51) 以降にその他の命令が載っている

配列

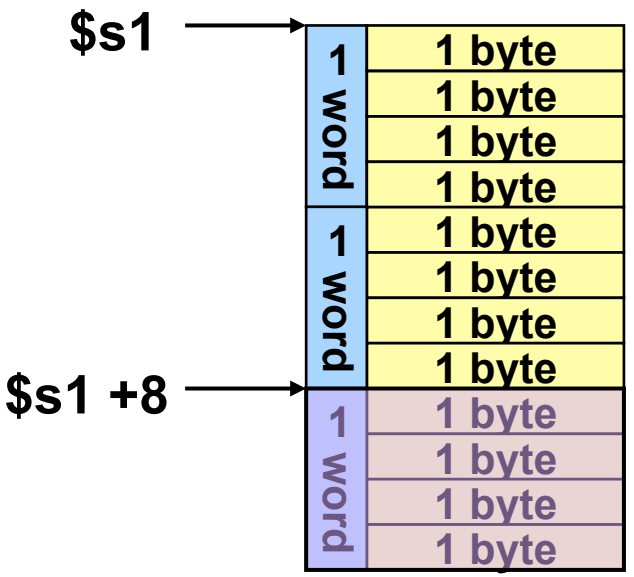
- ワードの配列
  - .word でワード (4 byte)の数値を定義できる
    - MIPSでは、1 ワード(語) = 4バイト = 32ビット
      - 1バイト = 8ビット
  - 要素 `A[i]` にはアドレス `A+i*4` が対応
    - ワード単位でデータは格納されるが、アドレッシングはバイト単位で表現

```
        .data
A:
        .word 1 2 3
```



メモリアクセス命令

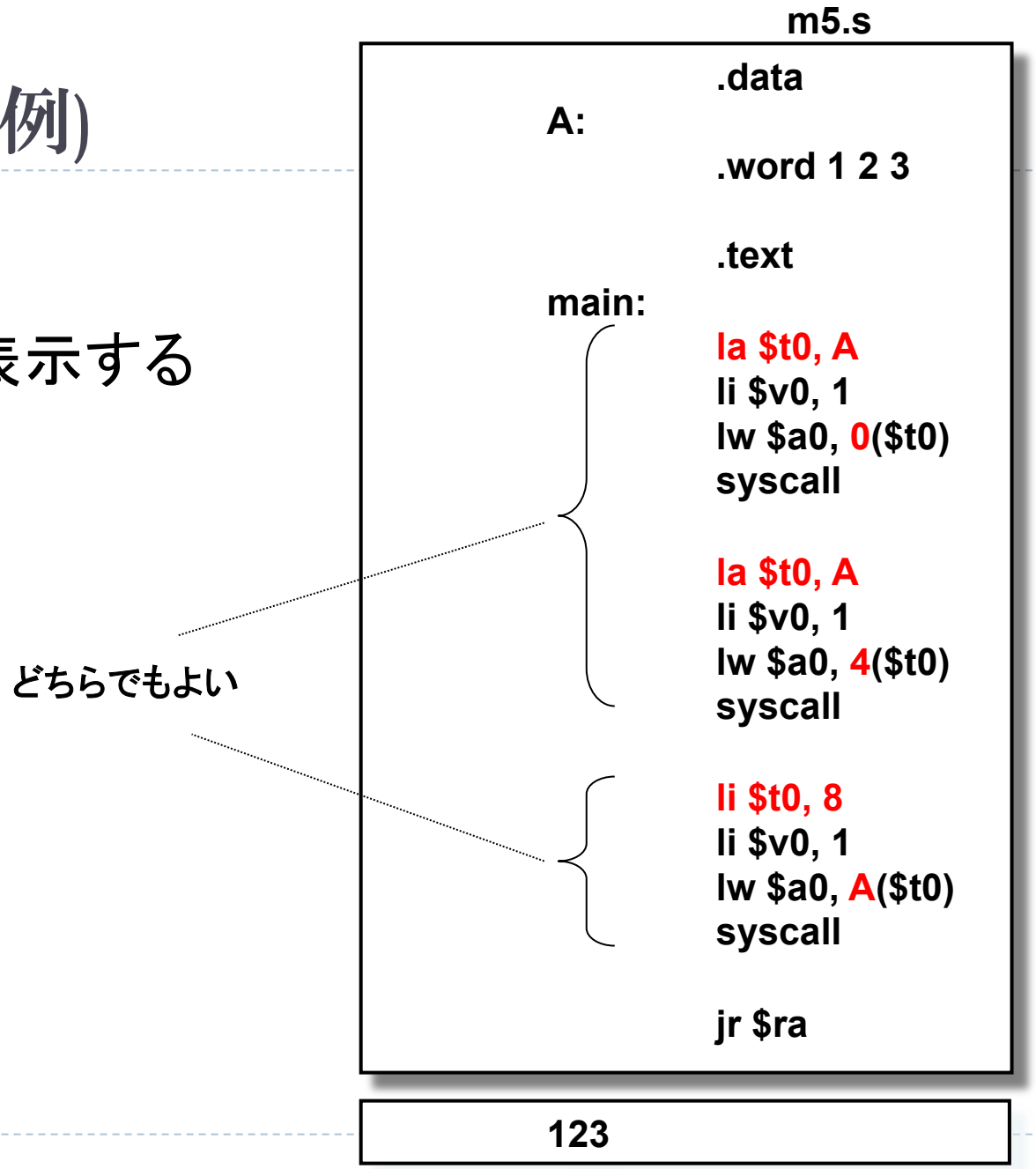
- lw $A, X($B)
  - $A = メモリ[X + $B]
    - $A, $B：レジスタ、X：定数 (ラベルor数字)
    - メモリ上のアドレス「X+$B」から始まる
      1ワード(4バイト)のデータを
      レジスタ$Aに転送する
  - 例：lw $t0, 8($s1)
    - $s1が示すアドレスから2ワード
      先のデータ(1ワード分)を$t0に読込む



- sw $A, X($B)
  - メモリ[X+$B] = $A
    - $A, $B：レジスタ、X：定数 (ラベルor数字)
    - レジスタ$Aの値 (1ワード)を
      メモリ上のアドレス「X+$B」に転送する
  - 例：sw $t0, 8($s1)
    - $s1が示すアドレスから2ワード先のデータ(1ワード分)に$t0の値を書き込む

配列の操作 (例)

- 配列Aの値を表示するプログラム



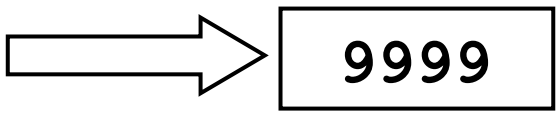
課題

課題1：bltの実装

- sltを用いてbltを実装せよ
  - 「blt　$s1, $s2, label」を対象に、一時レジスタとして$t0を使うこと
  - bltを実装したコード断片をレポートに記述し、その解説をすること
  - 注意点
    - 短いコードで記述されたもの程よい。

課題2: 2つの配列の要素の和

- これら2つの配列の i 番目と 3-i 番目の要素どうしを足し合わせ、表示せよ
- .wordとして、2つの配列を定義
  - A = { 1, 2, 3, 4 }
  - B = { 5, 6, 7, 8 }
- 注意点
  - A[0] + B[3], A[1] + B[2], A[2] + B[1], A[3] + B[0]
  - ループ処理で実装すること
  - syscallを使ってコンソールに表示すること



課題3：最大値の探索

- 配列の中から最大値を出力するコードをかけ
  - 結果をsyscallを使ってコンソールに出力すること

- 以下のような配列を定義
  - A = { 2, 7, 8, 1, 3, 9 }

補足

課題2

```
        .data
A:
        .word  1 2 3 4
B:
        .word  5 6 7 8

        .text
main:
        #和を表示
```

課題3

```
        .data
A:
        .word  2 7 8 1 3 9

        .text
main:
        #最大値を出力
```

課題提出

- 〆切: 5/21(月) 23:59
- 提出物: 以下のファイルをアーカイブ, 圧縮したもの
  - ドキュメント(pdf,plain txt,wordなんでも可)
    以下の内容を含めること
    - 課題1の解答
    - 課題2,3の実行結果
    - もしあれば感想、質問等
  - プログラムソース
    - 課題2,3 (適宜コメントを記述すること)