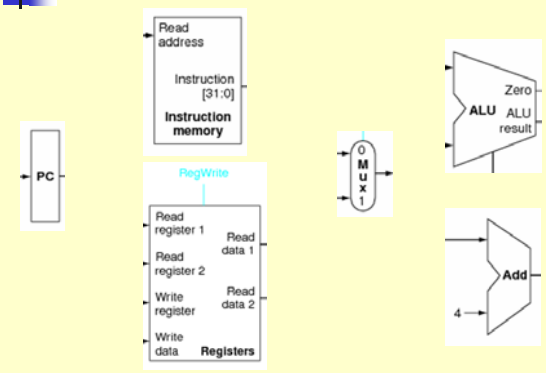
2010年 後学期

計算機アーキテクチャ 第二 (O)

3．RISCプロセッサとパイプライン処理

大学院情報理工学研究科 計算工学専攻
吉瀬謙二 kise _at_ cs.titech.ac.jp
S321講義室 月曜日 5，6時限 13：20－14：50

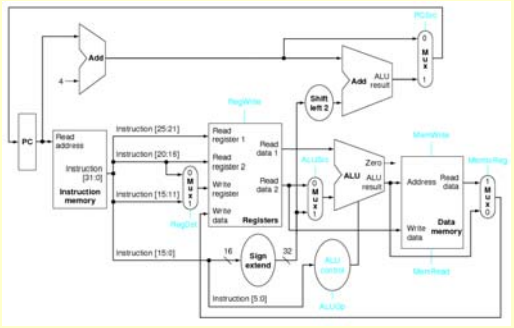
プロセッサの構成要素（1）



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]



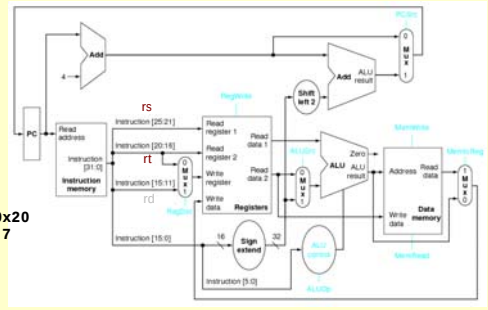
プロセッサの構成要素（2）

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]



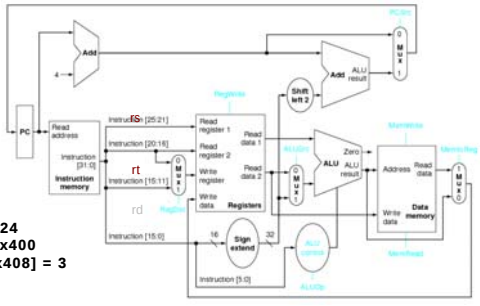
PC = 0x20
$29 = 7

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 $t0, 8($s2)　[lw $8, 8($18)]



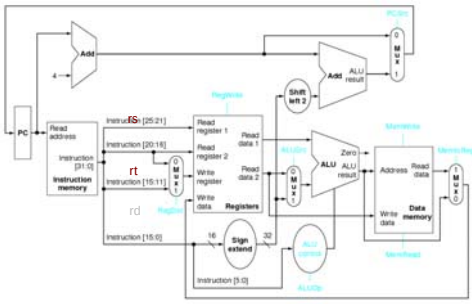
PC = 0x24
$18 = 0x400
mem[0x408] = 3

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)]

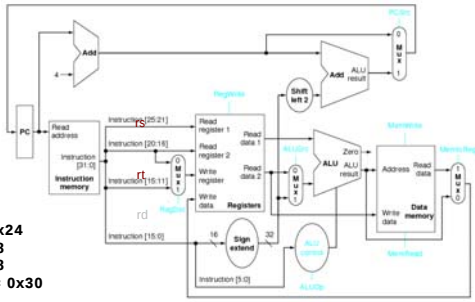


プロセッサのデータパス（シングル・サイクル）Exercise

op	rs	rt	16 bit immediate

I format

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



PC = 0x24
$16 = 8
$17 = 8
Label = 0x30

MIPS **Control Flow** Instructions

- MIPS conditional branch instructions:

```
bne $s0, $s1, Lbl #go to Lbl if $s0≠$s1
beq $s0, $s1, Lbl #go to Lbl if $s0=$s1
```

  - Ex:

```
if (i==j) h = i + j;
        bne $s0, $s1, Lbl1
        add $s3, $s0, $s1
Lbl1:   ...
```

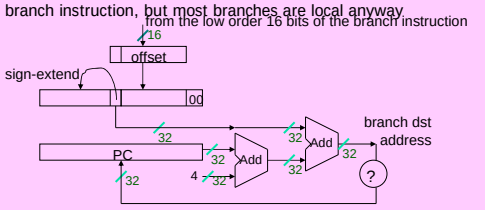
- Instruction Format (I format):

op	rs	rt	16 bit offset

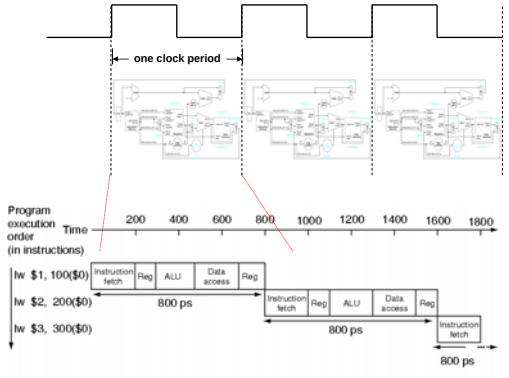
- How is the branch destination address specified?

Specifying Branch Destinations

- Use a register (like in lw and sw) added to the 16-bit offset
  - which register? Instruction Address Register (the PC)
    - its use is automatically implied by instruction
    - PC gets updated (PC+4) during the fetch cycle so that it holds the address of the next instruction
  - limits the branch distance to -2^{15} to $+2^{15}-1$ instructions from the (instruction after the) branch instruction, but most branches are local anyway



プロセッサのデータパス（シングル・サイクル）



simcore/mips

- C言語で記述したシンプルなMIPSプロセッサ
- プロセッサシミュレータ

test03

cat main.c

```
#include <stdio.h>
#include <stdlib.h>

int main(){
    int sum=0, i=0;
    for(i=0; i<=100; i++) sum+=i;
    return sum;
}
```

make

```
mipsel-linux-gcc -fno-delayed-branch -O0 -S main.c
mipsel-linux-as -EL -g -mips1 main.s -o main.o
mipsel-linux-ld -EL -Bstatic -entry=main -T linkerscript --oformat ecoff-littlem
ips main.o -o loop
mipsel-linux-strip loop
mipsel-linux-objdump -M gpr-names=numeric -d loop

loop:     file format ecoff-littlemips

Disassembly of section .text:

0000000000002000 <.text>:
    2000:       27bdfff8        addiu   $29,$29,-24
    2004:       afbe0010        sw      $30,16($29)
    2008:       03a0f021        move    $30,$29
    200c:       afc0000c        sw      $0,12($30)
    2010:       afc00008        sw      $0,8($30)
    2014:       afc00008        sw      $0,8($30)
    2018:       1000000a        b       0x2044
    201c:       00000000        nop
    2020:       8fc3000c        lw      $3,12($30)
    2024:       8fc20008        lw      $2,8($30)
    2028:       00000000        nop
    202c:       00621021        addu    $2,$3,$2
    2030:       afc2000c        sw      $2,12($30)
    2034:       8fc20008        lw      $2,8($30)
    2038:       00000000        nop
    203c:       24420001        addiu   $2,$2,1
    2040:       afc20008        sw      $2,8($30)
    2044:       8fc20008        lw      $2,8($30)
    2048:       00000000        nop
    204c:       28420065        slti    $2,$2,101
    2050:       1440fff3        bnez    $2,0x2020
    2054:       00000000        nop
    2058:       8fc2000c        lw      $2,12($30)
    205c:       03c0e821        move    $29,$30
    2060:       8fbe0010        lw      $30,16($29)
    2064:       27bd0018        addiu   $29,$29,24
    2068:       03e00008        jr      $31
    206c:       00000000        nop
```


test20

cat main.c

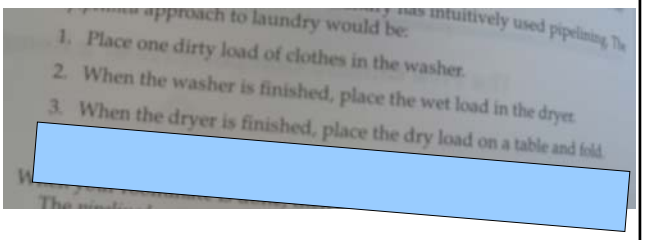
```
/*************************************************************/
typedef struct array_t{
    unsigned int cdt; /* candidates              */
    unsigned int col; /* column                  check */
    unsigned int pos; /* positive diagonal check */
    unsigned int neg; /* negative diagonal check */
} array;

/*************************************************************/
int main(){
    array a[32];
    int n = 7; /* problem size */
    int answers=0;

    int h = 1;
    int r   = (1<<n)-1;
    a[h].col = (1<<n)-1;
    a[h].pos = a[h].neg = 0;

    for(;;){
        if(r){
            int lsb = (-r) & r;
            a[h+1].cdt = (       r & ~lsb);
            a[h+1].col = (a[h].col & ~lsb);
            a[h+1].pos = (a[h].pos |  lsb) << 1;
            a[h+1].neg = (a[h].neg |  lsb) >> 1;

            r = a[h+1].col & ~(a[h+1].pos | a[h+1].neg);
            h++;
        }
        else{
            r = a[h].cdt;
            h--;

            if(r==0) break;
            if(h==n) answers++;
        }
    }

//    printf("N=%d, solutions %lld\n", n, answers);
    return answers;
}
```


計算機アーキテクチャ 第二 (O)

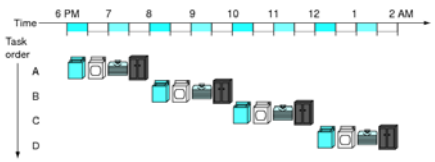
パイプライン処理

大学院情報理工学研究科 計算工学専攻
吉瀬謙二 kise _at_ cs.titech.ac.jp
S321講義室 月曜日 5，6時限 13:20－14:50

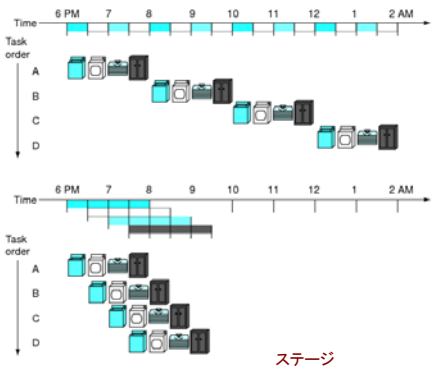
パイプライン処理 (pipelining)



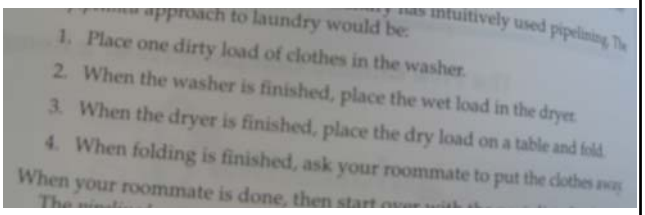
パイプライン処理 (pipelining)



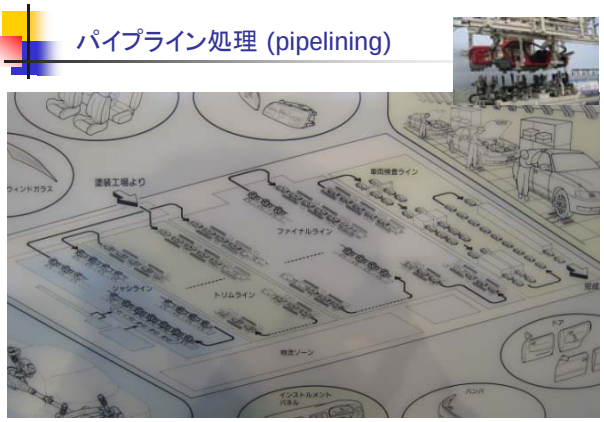
パイプライン処理 (pipelining)



パイプライン処理 (pipelining)



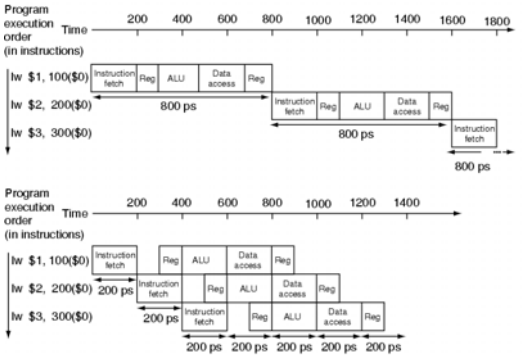
パイプライン処理 (pipelining)



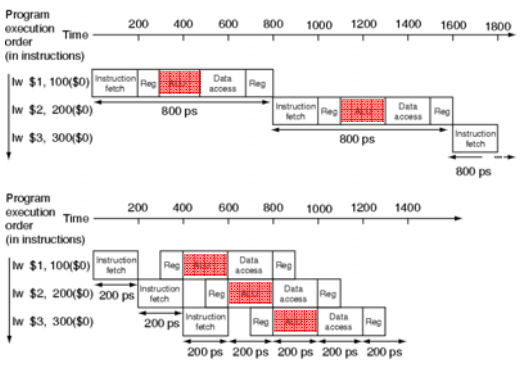
MIPSの基本的な5つのステップ（ステージ）

- **IF (Instruction fetch)ステージ**
  メモリから命令をフェッチする.
- **ID (Instruction decode and register file read) 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 (Execution or address calculation) 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 (Data memory access) 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 (Write back) ステージ**
  結果をレジスタに書き込む.

パイプライン処理 (pipelining)



パイプライン処理 (pipelining)



パイプラインによる速度向上

- パイプラインステージの数（段数）: **n**
- 実行する命令の数: **s**
- パイプライン化されたプロセッサのクロックを単位時間とする.
- 全命令が終了するまでの理想的なサイクル数
  - **n + s – 1**
- パイプラインを利用しないシングルサイクルのプロセッサ
  - **n * s**

レポート 問題

1. 1から100までの加算(test01)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. 加算のプログラム(test01)を simcore/mips で実行し，正しく実行されていることを説明せよ．
3. 実行された命令の頻度を測定するように，simcore/mips のソースコードを変更せよ．これを用いて，加算のプログラム(test01)の実行命令頻度を求め，グラフにせよ．（ロード・ストア命令，分岐・ジャンプ命令，それ以外，の3つに分類して実行頻度をグラフにする．）
4. 同様に，N-queen (test20) の実行頻度をグラフにせよ．
5. この課題の感想をまとめること．
6. 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

- **11月9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．

アナウンス

- 講義スライド，講義スケジュール
  - www.arch.cs.titech.ac.jp
- 講義用の計算機
  - 131.112.16.56 (情報工学科の演習室からは入れません)
  - ssh archo@131.112.16.56