計算機アーキテクチャ特論
(Advanced Computer Architectures)

2. スカラプロセッサ，スーパースカラプロセッサ

吉瀬 謙二 計算工学専攻
kise _at_ cs.titech.ac.jp www.arch.cs.titech.ac.jp
W831 講義室 木曜日 9:00 – 10:30

MIPS R3000 Instruction Set Architecture (ISA)

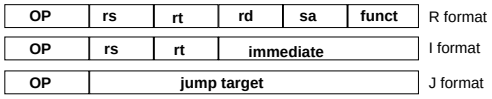
- Instruction Categories
  - Computational
  - Load/Store
  - Jump and Branch
  - Floating Point
    - coprocessor
  - Memory Management
  - Special

Registers

R0 - R31

PC
HI
LO

3 Instruction Formats: all 32 bits wide



MIPS Arithmetic Instructions

- MIPS assembly language **arithmetic statement**

```
add  $t0, $s1, $s2
sub  $t0, $s1, $s2
```

- Each arithmetic instruction performs only one operation
- Each arithmetic instruction fits in 32 bits and specifies exactly three operands

destination ← source1 op source2

- Operand order is fixed (destination first)
- Those operands are all contained in the datapath's register file ($t0,$s1,$s2) – **indicated by $**

MIPS ISA So Far

Category	Instr	Op Code	Example	Meaning
Arithmetic (R & I format)	add	0 and 32	add $s1, $s2, $s3	$s1 = $s2 + $s3
	subtract	0 and 34	sub $s1, $s2, $s3	$s1 = $s2 - $s3
	add immediate	8	addi $s1, $s2, 6	$s1 = $s2 + 6
	or immediate	13	ori $s1, $s2, 6	$s1 = $s2 v 6
Data Transfer (I format)	load word	35	lw $s1, 24($s2)	$s1 = Memory($s2+24)
	store word	43	sw $s1, 24($s2)	Memory($s2+24) = $s1
	load byte	32	lb $s1, 25($s2)	$s1 = Memory($s2+25)
	store byte	40	sb $s1, 25($s2)	Memory($s2+25) = $s1
	load upper imm	15	lui $s1, 6	$s1 = 6 * 2^{16}
Cond. Branch (I & R format)	br on equal	4	beq $s1, $s2, L	if ($s1==$s2) go to L
	br on not equal	5	bne $s1, $s2, L	if ($s1 !=$s2) go to L
	set on less than	0 and 42	slt $s1, $s2, $s3	if ($s2<$s3) $s1=1 else $s1=0
	set on less than immediate	10	slti $s1, $s2, 6	if ($s2<6) $s1=1 else $s1=0
Uncond. Jump (J & R format)	jump	2	j 2500	go to 10000
	jump register	0 and 8	jr $t1	go to $t1
	jump and link	3	jal 2500	go to 10000; $ra=PC+4

MIPS Register Convention, ABI (Application Binary Interface)

Name	Register Number	Usage	Preserve on call?
$zero	0	**constant 0 (hardware)**	n.a.
$at	1	reserved for assembler	n.a.
$v0 - $v1	2-3	returned values	no
$a0 - $a3	4-7	**arguments**	yes
$t0 - $t7	8-15	temporaries	no
$s0 - $s7	16-23	saved values	yes
$t8 - $t9	24-25	temporaries	no
$gp	28	global pointer	yes
$sp	29	stack pointer	yes
$fp	30	frame pointer	yes
$ra	31	return addr (hardware)	yes

Exercise 1

```
swap(int v[], int k)
{
  int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

Exercise 1

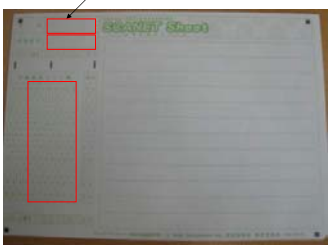
```
swap:
      add $t1, $a1, $a1  #
      add $t1, $t1, $t1  # $t1 = k * 4;
      add $t1, $a0, $t1  # $t1 = &v[k];

      lw  $t0, 0($t1)    # $t0 = v[k];
      lw                 #

      sw                 #
      sw                 #

      jr $ra             # return
```

sll $t1, $a1, 2

Exercise 2

```
swap(int v[], int j, int k)
{
  int temp;
  temp = v[j];
  v[j] = v[k];
  v[k] = temp;
}
```

氏名，学籍番号，
学籍番号マーク欄(右詰で)
アンケート番号 1



Exercise 2

```
swap:
      add $t1, $a1, $a1  #
      add $t1, $t1, $t1  # $t1 = k * 4;
      add $t1, $a0, $t1  # $t1 = &v[k];

      lw  $t0, 0($t1)    # $t0 = v[k];
      lw                 #

      sw                 #
      sw                 #

      jr $ra             # return
```

sll $t1, $a1, 2

Sample program

```
#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

Sample program

```
#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

```
main:
        .frame  $fp,24,$31
        .mask   0x40000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .set    nomacro

        addiu   $sp,$sp,-24
        sw      $fp,16($sp)
        move    $fp,$sp
        sw      $0,8($fp)
        li      $2,1
        sw      $2,12($fp)
        b       $L2
        nop

$L3:
        lw      $3,8($fp)
        lw      $2,12($fp)
        nop
        addu    $2,$3,$2
        sw      $2,8($fp)
        lw      $2,12($fp)
        nop
        addiu   $2,$2,1
        sw      $2,12($fp)
$L2:
        lw      $2,12($fp)
        nop
        slt     $2,$2,101
        bne     $2,$0,$L3
        nop

        lw      $2,8($fp)
        move    $sp,$fp
        lw      $fp,16($sp)
        addiu   $sp,$sp,24
        j       $31
        nop
```


Sample program

```
#include <stdio.h>

int main(){
  int i;
  int sum = 0;

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

  return sum;
}
```

mipsel-linux-objdump -d ./a.out

```
004005c0 <main>:
  4005c0:   27bdffe8      addiu   sp,sp,-24
  4005c4:   afbe0010      sw      s8,16(sp)
  4005c8:   03a0f021      move    s8,sp
  4005cc:   afc00008      sw      zero,8(s8)
  4005d0:   24020001      li      v0,1
  4005d4:   afc2000c      sw      v0,12(s8)
  4005d8:   1000000a      b       400604 <main+0x44>
  4005dc:   00000000      nop
  4005e0:   8fc30008      lw      v1,8(s8)
  4005e4:   8fc2000c      lw      v0,12(s8)
  4005e8:   00000000      nop
  4005ec:   00621021      addu    v0,v1,v0
  4005f0:   afc20008      sw      v0,8(s8)
  4005f4:   8fc2000c      lw      v0,12(s8)
  4005f8:   00000000      nop
  4005fc:   24420001      addiu   v0,v0,1
  400600:   afc2000c      sw      v0,12(s8)
  400604:   8fc2000c      lw      v0,12(s8)
  400608:   00000000      nop
  40060c:   28420065      slti    v0,v0,101
  400610:   1440fff3      bnez    v0,4005e0 <main+0x20>
  400614:   00000000      nop
  400618:   8fc20008      lw      v0,8(s8)
  40061c:   03c0e821      move    sp,s8
  400620:   8fbe0010      lw      s8,16(sp)
  400624:   27bd0018      addiu   sp,sp,24
  400628:   03e00008      jr      ra
  40062c:   00000000      nop
```

Sample program

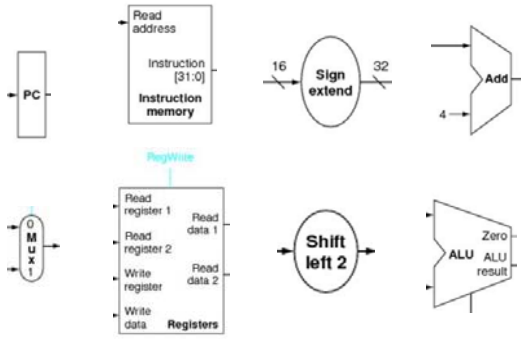
```
main:
        .frame  $sp,0,$31
        .mask   0x00000000,0
        .fmask  0x00000000,0
        .set    noreorder
        .set    nomacro

        j       $31
        li      $2,5050
```

```
# Makefile
all:
    mipsel-linux-gcc -O0 -S main.c -o main_opt0.s
    mipsel-linux-gcc -O1 -S main.c -o main_opt1.s
    mipsel-linux-gcc -O2 -S main.c -o main_opt2.s
    mipsel-linux-gcc -O3 -S main.c -o main_opt3.s
```


計算機アーキテクチャ特論
(Advanced Computer Architectures)

2-1. スカラプロセッサ

プロセッサの構成要素

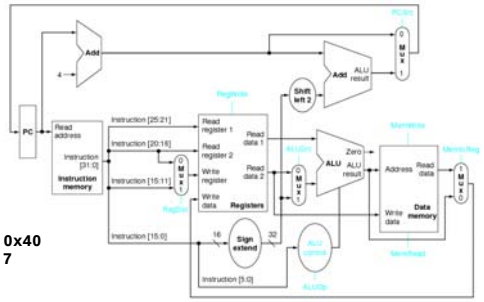


プロセッサの**データパス**(シングル・サイクル)

op	rs	rt	16 bit immediate

I format

addi $t0, $t1, -1 **[addi $8, $9, -1]**



PC = 0x40
$9 = 7

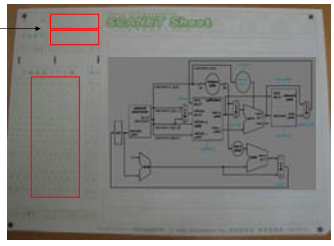
Exercise

op	rs	rt	16 bit immediate

I format

addi $t0, $t1, -1 **[addi $8, $9, -1]**

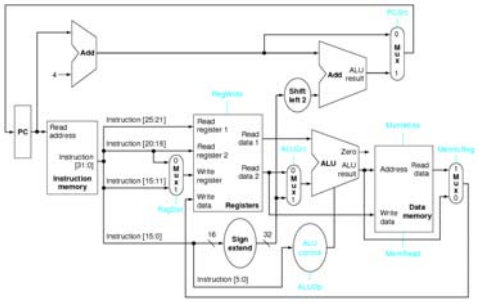
氏名，学籍番号，
学籍番号マーク欄(**右詰で**)



プロセッサの**データパス**(シングル・サイクル)

op	rs	rt	rd	shamt	funct

add $t0, $s1, $s2 **[add $8, $17, $18]**

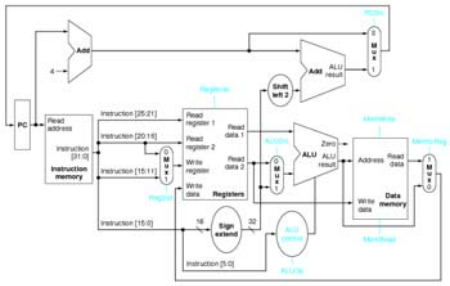


プロセッサのデータパス（シングル・サイクル）

op	rs	rt	16 bit immediate

I format

lw $t0, 24($s2)　[lw $8, 24($18)]

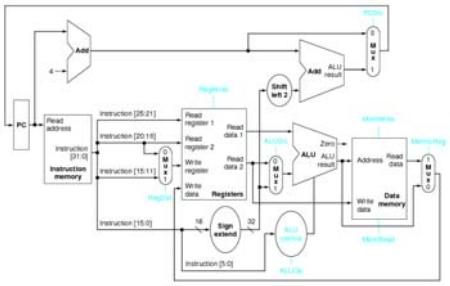


プロセッサのデータパス（シングル・サイクル）

op	rs	rt	16 bit immediate

I format

sw $t0, 24($s2)　[sw $8, 24($18)]

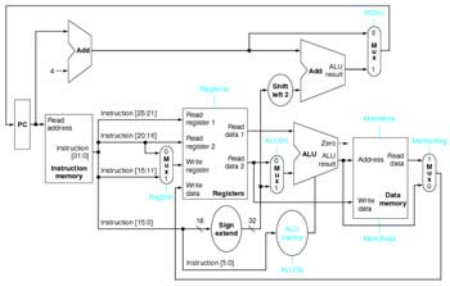


プロセッサのデータパス（シングル・サイクル）

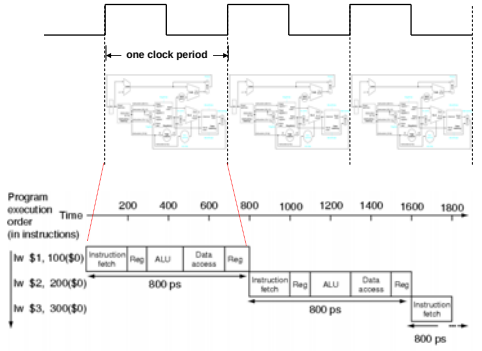
op	rs	rt	16 bit immediate

I format

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

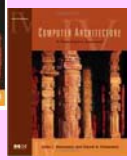


プロセッサのデータパス（シングル・サイクル）



教科書

- 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
- Computer Architecture, Fourth Edition: A Quantitative Approach, Fourth Edition
  - Publisher: Morgan Kaufmann; 4 edition (September 13, 2006)
  - ISBN-10: 0123704901
  - ISBN-13: 978-0123704900



参考書

- **参考書**
  - **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，村岡 洋一 著，近代科学社，1989
  - 計算機システム工学，富田 真治，村上 和彰 著，昭晃堂，1988
  - コンピュータハードウエア，富田 真治，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
- 教科書
  - 命令レベル並列処理，安藤秀樹 コロナ社

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 advance@131.112.16.56
    - username advance