

計算機構成論

Lecture 5

計算機の命令セット

2023年度春学期

情報理工学部 Rクラス担当

越智裕之

※ このレジュメの中で「教科書」とは、一昨年度まで教科書に指定されていた書籍のことです（Lecture 0 参照）。この書籍を入手できなくても支障なく学習できるよう、レジュメに加筆修正を行っています。

内容

MIPSの基本的な命令セットの理解

- MIPSの命令フォーマット

• C言語→アセンブリ言語→機械語

配布資料「MIPS命令早見表」
のみを見て自分で変換できるように

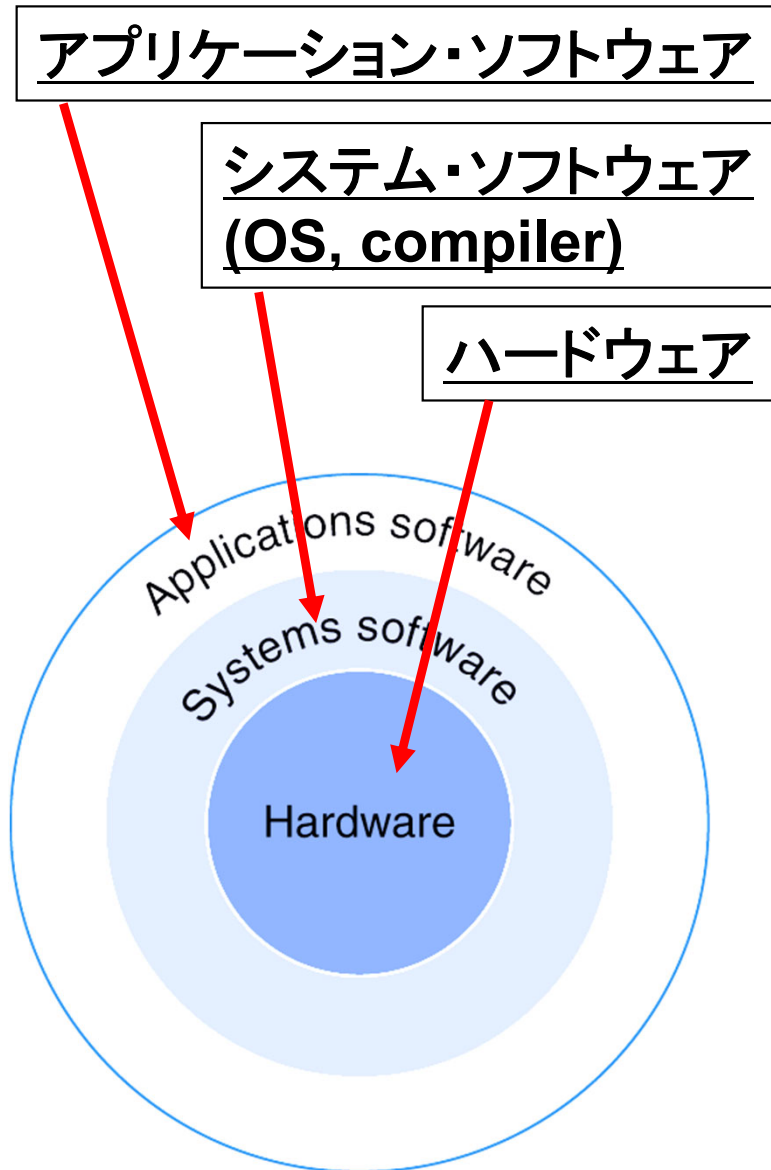
- 算術演算: $A = A + 4;$
- 論理演算: $A = A \ll 4;$ $A = A \& B;$ $A = \sim A;$
- 条件判定: $\text{If } (i == j) \text{ } f = g + h; \text{ else } f = g - h;$
- ループ: $\text{while } (i != j) \text{ } x = x + 1;$

- 32ビットの即値のオペランド
- MIPSのアドレッシング・モード

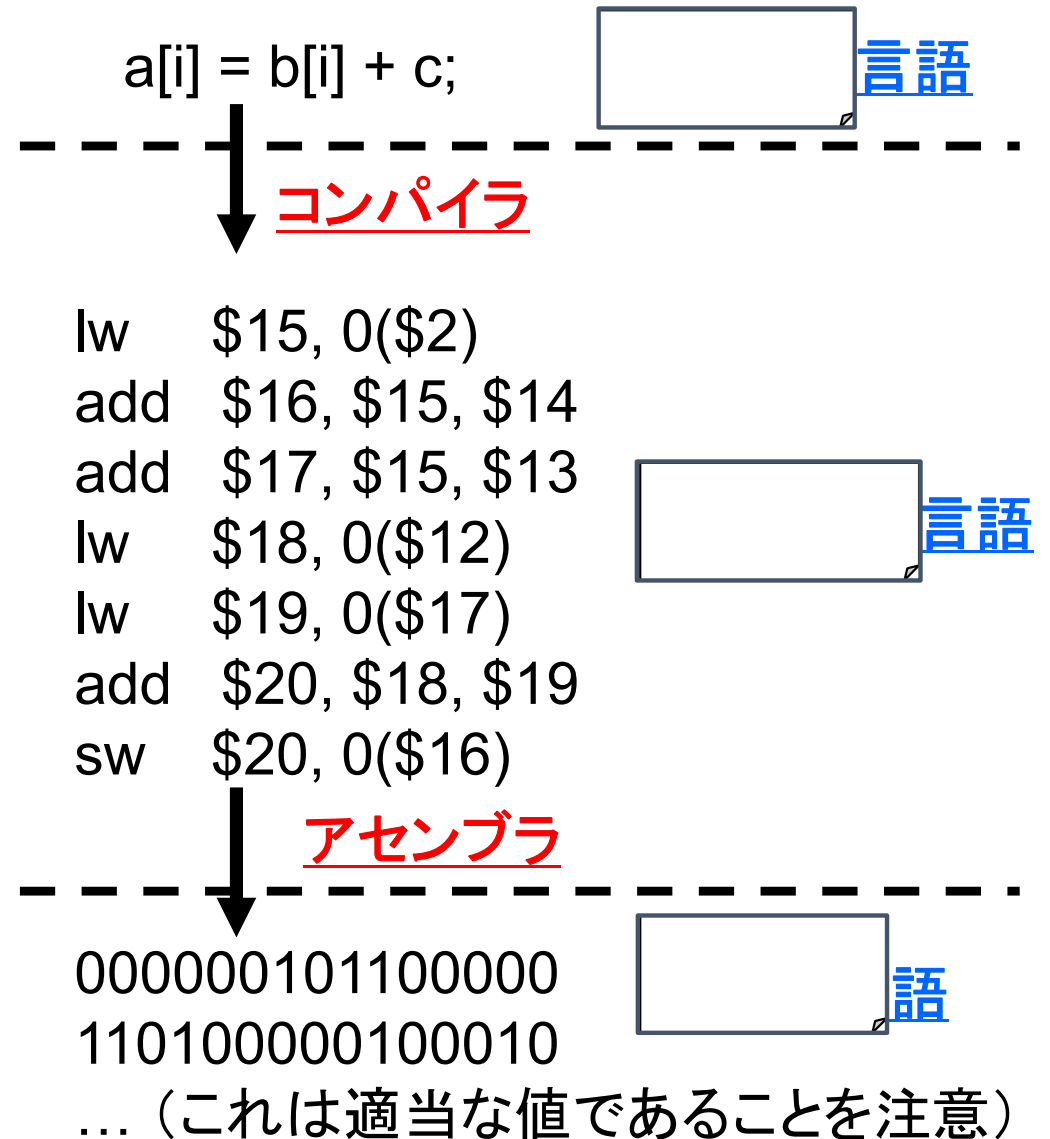
- 教材：教科書2.5, 2.6, 2.7, 2.10

C言語のプログラム実行の裏側

階層(抽象化)



下線つきの用語は全て理解すべし




ミニクイズ

A = 1010, B=1100 とする。簡単のため 4 ビットで考える。
では、以下のC値はどうなるか？

- ① C = A << 1;
- ② C = A >> 1;
- ③ C = ~A;
- ④ C = A & B;
- ⑤ C = A ^ B;

gccではsignedなら算術シフト、unsignedなら論理シフト

命令と命令セット

命令 (instruction) : コンピュータへの指示.
各コンピュータが理解できる命令の集まりを
と呼ぶ.

- 命令を並べたものがプログラム
- 命令セットはプロセッサによって異なる
- これからは, MIPS の命令セットで議論

命令セットの違い

例1：MIPS の加減算命令

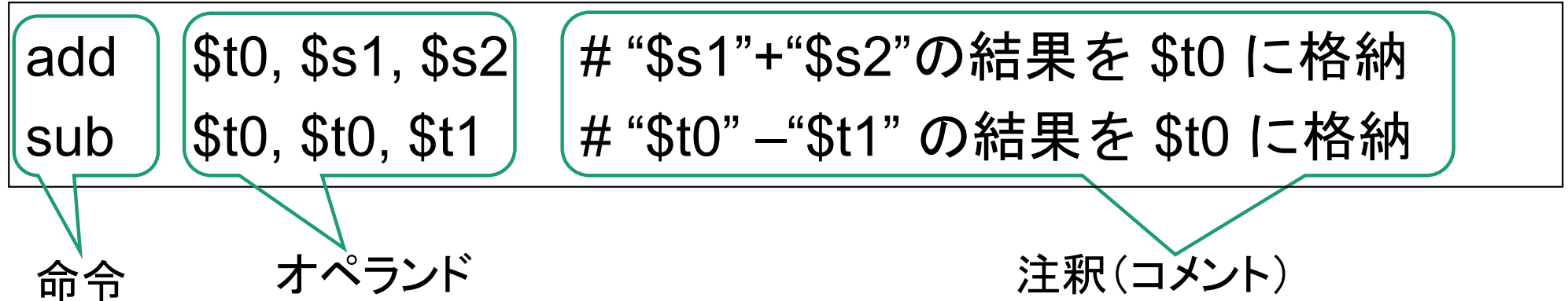
add	a, b, c	# $b + c$ の結果を a に格納
sub	a, b, c	# $b - c$ の結果を a に格納

例2：x86 の加減算命令

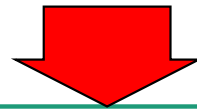
add	a, b	; $a + b$ の結果を a に格納
sub	a, b	; $a - b$ の結果を a に格納

MIPSの命令

MIPS の加減算命令



MIPS の算術／論理演算命令はどれも上記の3オペランド形式に限定



設計原則①： 単純性は規則性につながる。教科書 p 65

MIPSの命令形式 () は, 3つある

R形式、I 形式、J形式

R形式の命令(1/2)

add \$t0, \$s0, \$s1



000000 10000 10001 01000 00000 100000

1つの命令を32ビットに符号化して に格納

命令語に符号化すべき情報:

- 命令の種類: add
- ディスティネーションレジスタ: \$t0
- ソースレジスタ: \$s0, \$s1

(他の命令では) 定数やアドレスなどの情報もあり

*ディスティネーションやソースを英語でかけますね？

R形式の命令(2/2)

add \$t0, \$s0, \$s1

op	rs	rt	rd	shamt	funct
000000				00000	100000

add \$s0 \$s1 \$t0

加算または減算を意味する.

100000のときは加算,
100010のときは減算を意味する.

01000	\$t0	10000	\$s0
01001	\$t1	10001	\$s1
...
01111	\$t7	10111	\$s7

I形式の命令(1/2)

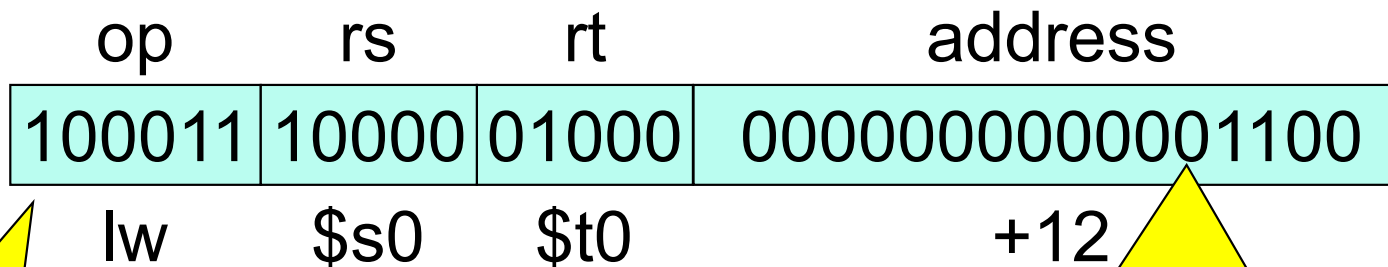
lw \$t0, 12(\$s0)

命令語に符号化すべき情報:

- 命令の種類: lw
- ディステーションレジスタ: \$t0
- ベースレジスタ: \$s0
- オフセット: 12
 - 2の補数によりマイナスも表現可能

I形式の命令(2/2)

lw \$t0, 12(\$s0)

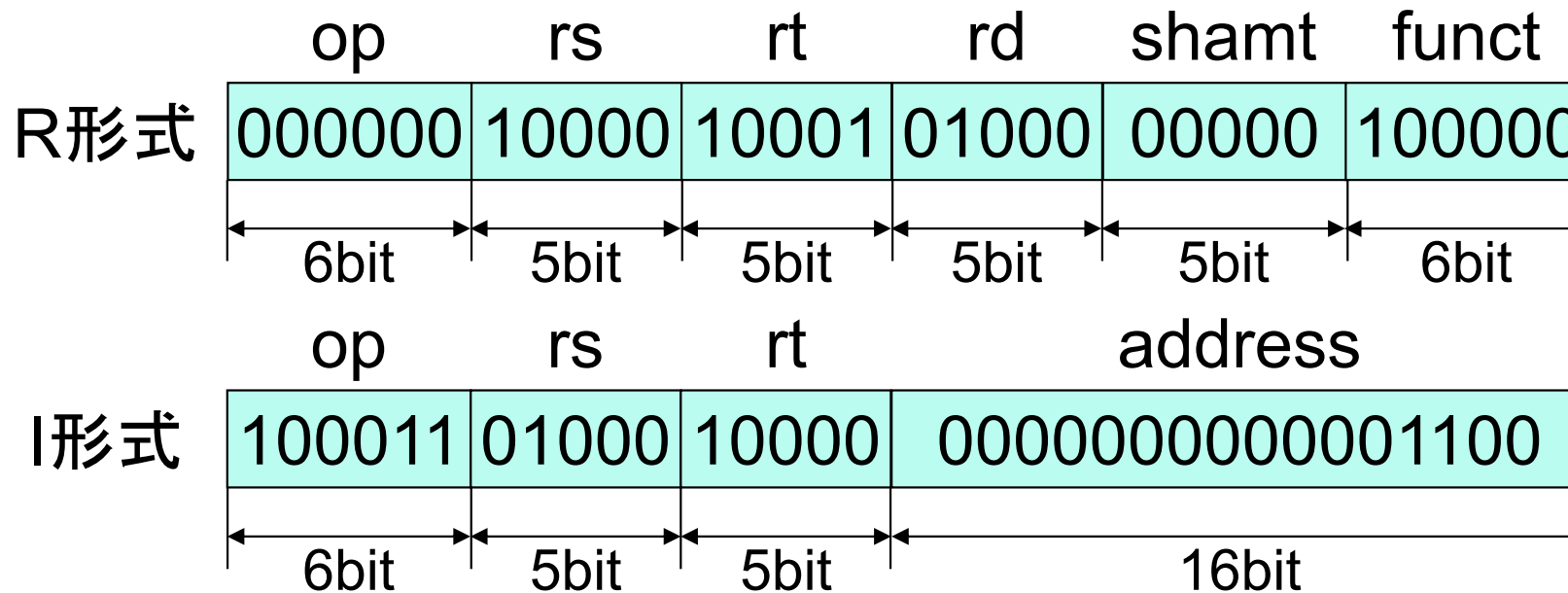


100011のときは lw,
101011のときは sw を
意味する.

16bit符号つき整数
(2の補数表現)
 $-32768(2^{15}) \sim +32767$

01000	\$t0	10000	\$s0
01001	\$t1	10001	\$s1
...
01111	\$t7	10111	\$s7

R形式とI形式の違い

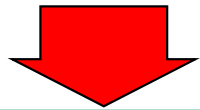


- MIPS は命令語を32bit 幅で統一している.
- MIPS の命令形式は, R形式, I形式, J形式 (後述) の3種類がある.
- どの命令形式かは, op フィールド (命令操作コード) で判別できる.

MIPSの設計思想

命令形式設計のポイント:

- 命令形式の種類は少なく!
- 命令形式には規則性を!



ハードウェアの単純化と高速化

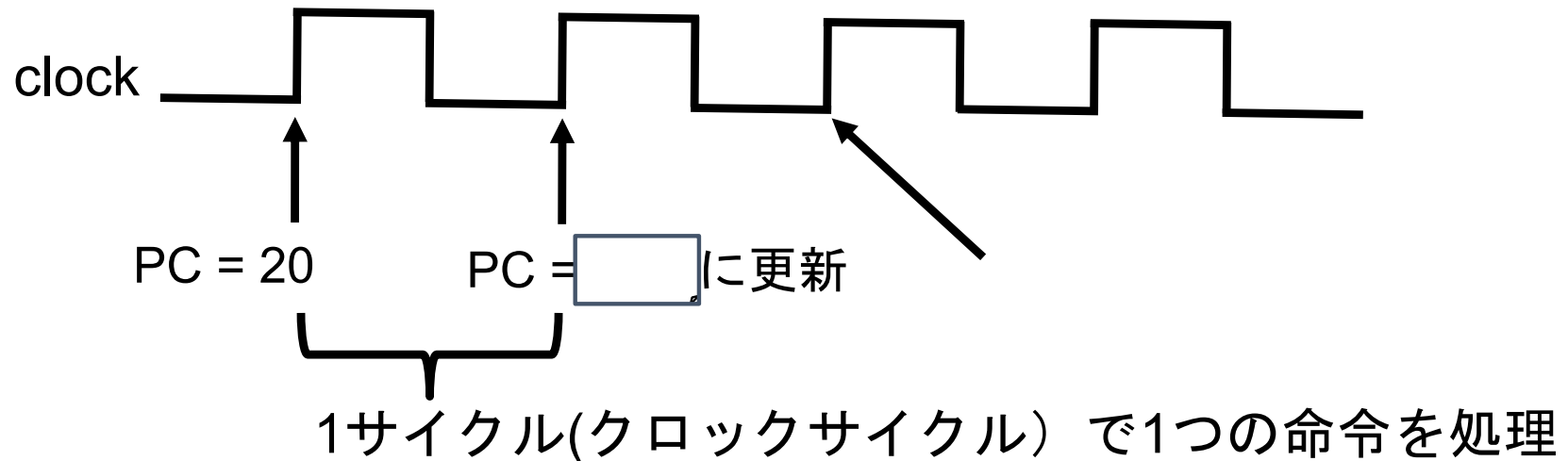
ベースレジスタの番地より 16ビットの情報で表現できる以上遠くのアドレスにアクセスすることはあきらめる

設計原則④(p88): すぐれた設計には適度な妥協が必要

* 以下の動作イメージの概念がわかると今後の計算機の挙動が理解しやすいはず

重要

すべてのレジスタはクロックの立ち上がりで更新とする



20番地に書かれた命令を実行

- このサイクルの最後に実行した命令で必要なレジスタは更新
- このサイクルの最後にPCを更新
 - jやbeqでなければPCを の値に更新
 - jやbeqの場合はPCを必要な値に変更

* この講義スライドの中のほとんどの数字は10進数です.

J形式の命令(1/2)

j Label

命令語に符号化すべき情報:

- 命令の種類: j
- Jumpするアドレス(ラベルからアセンブラが計算して埋め込む、後述)

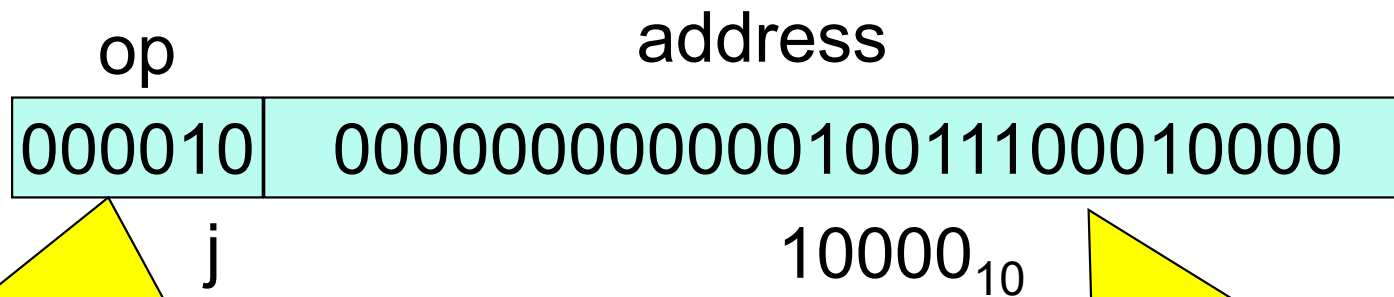
```
Loop: sll    $t1, $s3, 2
      add    $t1, $t1, $s6
      lw     $t0, 0($t1)
      slt    $s1, $t0, $s5
      bne    $s1, $zero, Exit
      addi   $s3, $s3, 1
      j      Loop
```

```
Exit:
```

J形式の命令(2/2)

jLabel

ラベルのアドレスが 40000_{10} のときは以下の機械語となる



000010のときは j, 000011のときは jal を意味する.

26bitのアドレス

- 26ビットで28ビット分の情報を表せる (Why?次ページ)
- PCの下位28ビットをaddressで置き換え(次頁以降を参照)
- $2^{28}=256\text{M}$ バイトの境界を越えてJumpするには？(演習⑨-(2))

26ビットで28ビット分の情報を表せるのはなぜか？

重要なポイント：

PCの値は4の倍数のため、下位2ビットは必ず00


そのため、その00を覚える必要はない

Jump命令の補足 (1/3)

- PC: Program Counter の略

—  を保持

MIPSの命令は32ビットなので、現在PCの値が2048なら、次の命令の格納されている番地は？

そのため、通常は1命令実行時に、並列して $PC = PC +$  という動作を（命令に関係なく自動で）行っている

しかし、j命令がくると、その指定した28ビットでPCの下位28ビットを置き換える

Jump命令の補足 (2/3)

[illegible]

番地の命令がJump系の命令でない時

あるサイクルで

[illegible]

このサイクルでやることは、

- [illegible]

00000000000000000000000000000000

次のサイクルでは、上記のPCの値で同じことをする

Jump命令の補足 (3/3)

[illegible]

26ビット

あるサイクルで

PC = 0000000000000000000000000000000010100 とする。

このサイクルでやることは、

- [illegible]

[illegible]

0110000000000000000000000000000010100 命令に埋め込まれた26ビット+00

00000000000000000000000000000000000011000:ジャンプしない時のPC

*このスライドのカラーは対応しています。講義の時か配布スライドで色を確認しましょう。

Jump 自己確認復習：■に入る2進数は？

[illegible]

■ は26ビットで下線部の1/4の値を入れる

I形式の別の命令

beq \$s0, \$t0, Label

op	rs	rt	address
000100	10000	01000	000000000000001100
beq	\$s0	\$t0	+12

16bit符号つき整数
(2の補数表現)
-32768(2^{15})~
+32767

if (\$s0 == \$t0)

PC = (PC+4) + 12×4

01000	\$t0	10000	\$s0
01001	\$t1	10001	\$s1
...
01111	\$t7	10111	\$s7

(自分で) 配布資料からbneの意味と機械語を確認して置いてください。
(試験にbneが出ても文句言わないように)

内容

MIPSの基本的な命令セットの理解

- MIPSの命令フォーマット

- C言語→アセンブリ言語→機械語

- 算術演算: $A = A + 4;$
 - 論理演算: $A = A \ll 4;$ $A = A \& B;$ $A = \sim A;$
 - 条件判定: $\text{If } (i == j) \text{ } f = g + h; \text{ else } f = g - h;$
 - ループ: $\text{while } (i != j) \text{ } x = x + 1;$

配布資料「MIPS命令早見表」
のみを見て自分で変換できるように

- 32ビットの即値のオペランド
- MIPSのアドレッシング・モード

- 教材：教科書2.5, 2.6, 2.7, 2.10

教科書P.70の例

以下のCのコードをMIPSのアセンブリ言語に直せ.
変数はすべてint型とする.

`A[12] = h + A[8];`

ただし、変数 `h` はレジスタ `$s2` に割り付けられており、
配列 `A` の先頭アドレスはレジスタ `$s3` に格納されているとする.

答え :

```
lw    $t0, ($s3)    # A[8] is brought into $t0
add   $t0, $s2, $t0          # h + A[8] is in $t0
sw    $t0, ($s3)    # $t0 is stored into A[12]
```


演習問題 その①

以下のCのコードをMIPSのアセンブリ言語に直せ.
変数はすべてint型とする.

`d[3] = d[2] + a;`

ただし、`a`が格納されているアドレスは`$s1`, `d`の先頭アドレスは、`$s4`に格納されているとする.

(前頁の例より, ちょっとだけ難しい)

答え :

```
lw    $t0,     # d[2] is brought into $t0
lw    $t1,     # a is brought into $t1
add       # the sum is in $t0
sw    $t0,     # $t0 is stored into d[3]
```

Fact: アセンブリ言語に変換するとサイズが増大

演習問題 その②

制限時間5分程度

演習問題 その①のアセンブリ言語を機械語に直せ

	op	rs	rt	address
lw \$t0, 8(\$s4)	100011	10100	01000	000000000000001000
	lw	\$s4	\$t0	+8

lw \$t1, 0(\$s1)

add \$t0, \$t0, \$t1

sw \$t0, 12(\$s4)

内容

MIPSの基本的な命令セットの理解

- MIPSの命令フォーマット

- C言語→アセンブリ言語→機械語

配布資料「MIPS命令早見表」
のみを見て自分で変換できるように

- 算術演算: $A = A + 4;$
- 論理演算: $A = A \ll 4;$ $A = A \& B;$ $A = \sim A;$
- 条件判定: $\text{If } (i == j) \text{ } f = g + h; \text{ else } f = g - h;$
- ループ: $\text{while } (i != j) \text{ } x = x + 1;$

- 32ビットの即値のオペランド

- MIPSのアドレッシング・モード

教科書2.10節

- 教材：教科書2.5, 2.6, 2.7, 2.10

即値のオペランド (Immediate Operands)

addi \$s0, \$s1, 1000 (10進数) に関してミニクイズ

1. これはどんな動作をするか？

配布資料を参考に

2. 1000の部分は何と呼ばれているか？

3. 1000の部分は何ビットの数が指定可能か？

使用する定数は通常そんなに大きくないが、もし32ビットの定数を(命令に埋め込んで)使いたい場合はどうするか？

32ビットの命令の中に32ビットの即値を埋め込めるわけがない！

そこで、定数を16ビットずつ命令に埋め込んでおき、2命令かけて32ビットの定数をレジスタにロードする方法がある(演習③)

MIPSのアドレッシング・モード (教科書p113)

今まで見てきた命令では、命令によってそのオペランドの解釈の仕方が異なる。

つまり、

- 命令が操作（addなど）の**対象とするもの**の決め方
 - 命令が必要とする（jumpなど）**アドレス**の決め方
- が、命令によって異なる。その方法を
アドレッシング・モードと呼び、以下の5通りある。

* 下線が付いている用語は、試験に出るかもしれない

1. レジスタ・アドレッシング
2. ベース相対アドレッシング
3. 即値アドレッシング
4. PC相対アドレッシング
5. 疑似直接アドレッシング

①レジスタ・アドレッシング

配布資料をよく見て
自分で一度確認して下さい

「指定したレジスタの中身」をオペランドにする

add \$t0, \$s0, \$s1

op	rs	rt	rd	shamt	funct
000000	10000	10001	01000	00000	100000
add	\$s0	\$s1	\$t0		

② ベース相対アドレッシング

「（指定したレジスタの中身+定数）の番地のメモリの内容」をオペランドにする

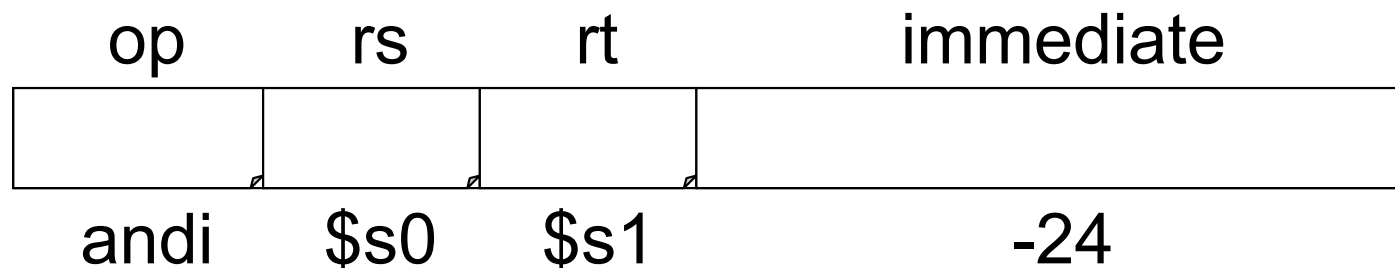
lw \$t0, 12(\$s0)

op	rs	rt	address
100011	10000	01000	000000000000001100
lw	\$s0	\$t0	+12

③即値アドレッシング

「指定した定数」をオペランドにする

```
andi $s1, $s0, -24
```



* 16ビットの即値は32ビットに拡張してから使用
この時、拡張のやり方に注意！（演習③参照）

* 配布資料をみて空欄が自分ですぐに埋められるように

④PC相対アドレッシング

「 $(PC+4) + (\text{指定した定数} \times 4)$ 」のアドレスを示す。

bne \$t0, \$s0, Label

op	rs	rt	address
000101	01000	10000	000000000000001100
bne	\$t0	\$s0	+12

自己確認のための重要なミニクイズ

上のbne命令が格納されているアドレスが300の時、Labelの指している命令が格納されているアドレスは何か？

⑤疑似直接アドレッシング

「（PCの上位4ビット）と（指定した定数×4）の接続」のアドレスを示す。

op	address
000010	00000000000010011100010000
j	

PC=00101100000000000010011100010000
のとき、上記の命令を実行した後のPCは？

PC=00101100000000000010011100010000

演習問題 その③

各自でやってください

32ビットの即値のオペランド (p109の例題)

次の32ビットの定数を\$s0にロードするMIPSのアセンブリ・コードを示せ
(定数をメモリに置かずに命令中に埋め込む)

0000 0000 0011 1101 0000 1001 0000 0000

61 2304

左の32ビットの定数を
16ビットずつに分けて
2命令に埋め込む

以下の命令の後で、\$s0がどうなるか自分で考えてみよう

lui (load upper immediate) 命令は「16ビットの即値定数フィールドの値」を指定されたレジスタの上位16ビットに格納し、そのレジスタの下位16bitにゼロを埋める

lui \$s0, 61

lui 命令で\$s0に
格納される値

0000 0000 0011 1101 0000 0000 0000 0000

ori \$s0, \$s0, 2304

ori 命令の即値
オペランド

0000 0000 0000 0000 0000 1001 0000 0000

ori 命令で\$s0に
格納される値

0000 0000 0011 1101 0000 1001 0000 0000

ori 命令のような**論理演算**命令の16bit即値は上位16bitに0を埋めて32bit化される

問①: 上の2命令目を `addi $s0, $s0, 2304` にすると結果はどうなるか?

問②: `addi $s0, $s0, X` と `ori $s0, $s0, X` の実行結果が異なるときの

Xの条件は? **addi** 命令のような**算術演算**命令の16bit即値は符号拡張され32bit化される

解答: ① , ② のとき

演習問題 その④

自分でやってください

教科書2.6章

以下のCのコードをMIPSのアセンブリ言語に直せ

① $C = A \ll 4;$

② $C = A \& B;$

③ $C = A | B;$

④ $C = \sim A;$

このC言語の意味は
覚えておくように
(=試験では注釈を
出しません)

ただし、Aが格納されているのは\$t1, Bが格納されているのが\$t2,
Cを格納するのが\$t0とする。

また、変換したアセンブリ言語を機械語にも直せ。

ヒント：MIPSにNOT演算はないので、NOR演算を使う。

ヒント：レジスタ\$zeroには定数0が入っている。

確認クイズ：以下の時の①～④の結果は？

A=0000 0101

B=1110 1011

演習問題 その④ (解答) R形式の機械語: 配布資料をみて変換できるように

①sll \$t0, \$t1, 4	op	rs	rt	rd	shamt	funct
	000000	00000	01001	01000	00100	000000
		0	\$t1	\$t0	4	0
②and \$t0, \$t1, \$t2	op	rs	rt	rd	shamt	funct
	000000	01001	01010	01000	00000	100100
		\$t1	\$t2	\$t0		36
③or \$t0, \$t1, \$t2	op	rs	rt	rd	shamt	funct
	000000	01001	01010	01000	00000	100101
		\$t1	\$t2	\$t0		37
④nor \$t0, \$t1, \$zero	op	rs	rt	rd	shamt	funct
	000000	01001	00000	01000	00000	100111
		\$t1	\$zero	\$t0		39

※ sll や srl や sra 命令は rs フィールドを使用しないので、機械語では 0 にしておく
 ※ sll や srl や sra を除く R形式の命令は shamt フィールドを使用しないので、機械語では 0 にしておく

演習問題 その⑤ in English

問題が英語でも解けるように
自分で穴埋め考えてください

Convert to assembly:

```
while (save[i] == k)
    i += 1;
```

i and k are in \$s3 and \$s5 and
base of array save[] is in \$s6

教科書p90

特に断りがなければ,
変数は全てint型とする.

以下のようなコメントは
試験問題ではつけない予定

Ans. Loop: sll \$t1,
add \$t1,
lw \$t0,
bne , Exit
addi
j Loop

Exit:

#最初の2行で、save[i]の
#アドレスを\$t1に
#\$t0にsave[i]の値を
#whileのbreak条件のチェック
i += 1

演習問題 その⑥

前の問題とほぼ同じだが、`==` が `<` に変わる場合、実行速度は同じだろうか、ちがうだろうか？

Convert to assembly:

`while (save[i] < k)` `i` and `k` are in `$s3` and `$s5` and
 base of array `save[]` is in `$s6`
 `i += 1;`

Ans.

Loop:	<code>sll</code>		#最初の2行で、 <code>save[i]</code> の
	<code>add</code>		# アドレスを <code>\$t1</code> に
	<code>lw</code>		# <code>\$t0</code> に <code>save[i]</code> の値を
	<code>slt</code>		#whileのbreak条件のチェック
	<code>beq</code>		#break条件成立時はExitに飛ぶ
	<code>addi</code>		# <code>i += 1</code>
	<code>j</code>		
Exit:			

なぜ「<を判定してjump」を1命令でできないか？

発展

while (save[i] == k)の場合
1命令で条件チェック&分岐

```
bne $t0, $s5, Exit
```

while (save[i] < k)の場合
2命令で条件チェック&分岐

```
slt $s1, $t0, $s5  
beq $s1, $zero, Exit
```

なぜ1命令に(MIPSを設計)しなかったのか？

<の判定は(==に比べて)時間がかかるので

たとえば、

- == の判定 10ns,
- PCの値の設定10ns,
- <の判定20ns

という場合を考えてみよう.

「複雑な命令を実装するよりも、速度の速い2つの命令を組み合わせた方がいいと判断したから (教科書p82)

符号付きの比較と符号なしの比較

符号付き／なしで、大小関係が変わる → それぞれに対応した比較命令

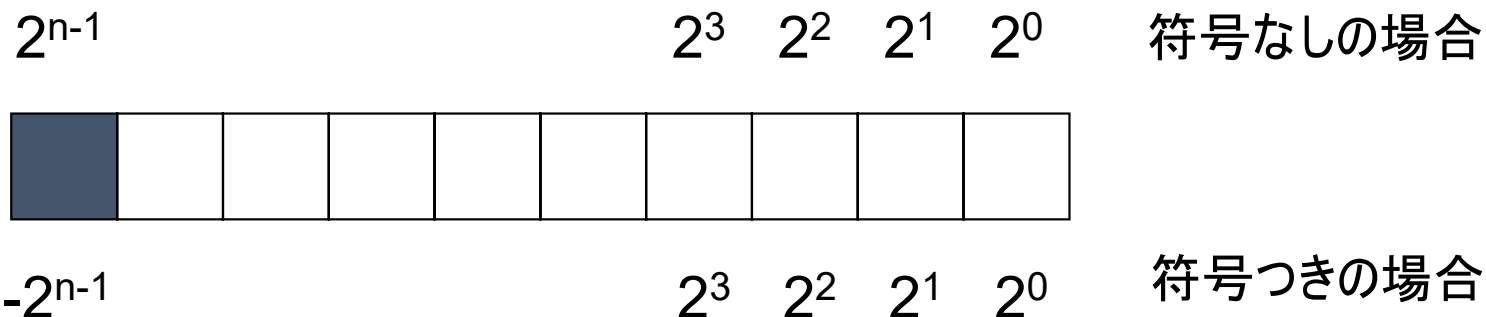
MIPSの
比較命令

符号付き整数用 比較命令 (Cでいえばint)	slt	(set on less than)
	slti	(set on less than immediate)
符号なし整数用 比較命令 (Cでいえばunsigned int)	sltu	(set on less than)
	sltiu	(set on less than immediate)

教科書
P92

slt	\$t0, \$s0, \$s1	#	符号付きの比較
sltu	\$t1, \$s0, \$s1	#	符号なしの比較

Lec. 3 覚えていますか？



確認クイズ

レジスタ\$s0に1111 1111 1111 1111 1111 1111 1111 1111₂

レジスタ\$s1に0000 0000 0000 0000 0000 0000 0000 0001₂

が格納されているとき、次の2つの命令で、レジスタ\$t0,\$t1の値はそれぞれ、10進数でどうなるか

```
slt    $t0,$s0,$s1    # 符号付きの比較
sltu   $t1,$s0,$s1    # 符号なしの比較
```

答)

\$s0の値：符号つき整数だと , 符号なし整数では 4,294,967,295₁₀
\$s1の値：符号つき整数だと , 符号なし整数では 1₁₀

符号付きの比較(slt)だと、\$s0<\$s1 → レジスタ\$t0の値は となる
符号なしの比較(sltu)だと、\$s0>\$s1 → レジスタ\$t1の値は となる

Lec. 5での要チェック用語集

命令

命令セット

MIPS

オペランド

コメント

ディスティネーションレジスタ

ソースレジスタ

ベースレジスタ

オフセット

命令操作コード

$C = A \ll 4;$

$C = A \& B;$

$C = A \mid B;$

$C = \sim A;$

アドレッシング・モード

演習問題 その⑦

教科書p89

以下のCのコードをMIPSのアセンブリ言語に直せ

```
if (i == j)
    f = g+h;
else
    f = g-h;
```

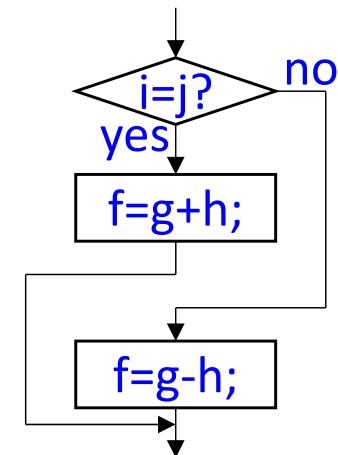
ただし、f, g, h, i, jは、\$s0から\$s4の5つのレジスタに割り付けられている

答え

if – else の構造はこう書ける

以下の模範解答の空欄を埋めよ。

```
        bne      [ ] Else
        add      [ ]
        j        Exit
Else:    sub      [ ]
Exit:
```



演習問題 その⑧=教科書 p112の例題

右のMIPSのアセンブリ・コードの格納されているアドレスが80000からのとき、このループの機械語はどうなるか？

数値は10進数であることに注意

```

Loop: sll    $t1, $s3, 2
      add    $t1, $t1, $s6
      lw     $t0, 0($t1)
      bne    $t0, $s5, Exit
      addi   $s3, $s3, 1
      j      Loop
    
```

Exit:

回答	80000	0	0	19	9	2	0
	80004	0	9	22	9	0	32
	80008	35	9	8	0		
	80012						
	80016						
	80020	2					
	80024						

配布資料（MIPS命令早見表）があれば、すぐに回答できるように

演習問題 その⑨

beq \$s0, \$s1, L1

で分岐先のL1のアドレスはこの命令のあるアドレス+4の前後
 $-2^{17} \sim +(2^{17}-4)$ の範囲にないといけない
(これがわからない場合は今までを要復習！)

(1) L1のあるアドレスがもっと遠い場合、ただし、
 $-2^{27} \sim +(2^{27}-4)$ の範囲にある場合はどうしたらいいか
(教科書 p113の例題と同じ)

(2) MIPSではプログラムは通常28ビットのアドレス空間
に配置されるので、(1)で十分である。しかし、特別な状
況で、L1のあるアドレスがさらにもっと遠い場合、飛び先
には32ビットのアドレスを指定しないといけない。そのア
ドレスが\$t0に入っているとすると、どうしたらいいか？
ヒント：jr \$t0 でPCを\$t0の値に設定できる。

演習問題 その⑩

演習問題 その⑧の機械語のコード（以下に再掲）の各命令を実行した直後のPCの値をそれぞれ答えよ。
bne命令については条件が成立した場合としなかった場合の両方を答えよ。

80000	0	0	19	9	2	0
80004	0	9	22	9	0	32
80008	35	9	8	0		
80012	5	8	21	2		
80016	8	19	19	1		
80020	2	20000				
80024						