東北大学 工学部 機械知能・航空工学科
2020年度 クラス C D

情報科学基礎 I

6. MIPSの命令と動作 — 演算・ロード・ストア

(教科書6.3節，6.4節
命令一覧は p.113)

大学院情報科学研究科
鏡 慎吾

MIPSのレジスタ間演算命令（復習）

復習: レジスタ間の演算命令

(C言語)

```
z = x & y;
```

ただし，変数 x, y, z の内容がそれぞれレジスタ s0, s1, s2 に置かれているとする

(MIPSアセンブリ言語)

```
and $s2, $s0, $s1      # $s1 ← $s0 & $s1
```

再掲: 主なレジスタ間演算命令

命令	説明
addu $c, $a, $b	$c ← $a + $b (add unsigned の略)
subu $c, $a, $b	$c ← $a – $b (subtract unsigned の略)
and $c, $a, $b	$c ← $a & $b
or $c, $a, $b	$c ← $a | $b
nor $c, $a, $b	$c ← ~($a | $b)
xor $c, $a, $b	$c ← $a ^ $b
sll $c, $a, $b	$c ← $a << $b (shift left logical の略)
srl $c, $a, $b	$c ← $a >> $b (shift right logical の略)
slt $c, $a, $b	符号つきで $a < $b ならば $c ← 1; さもなくば $c ← 0 (set on less than の略)
sltu $c, $a, $b	符号無しで $a < $b ならば $c ← 1; さもなくば $c ← 0 (set on less than unsigned)

再掲: レジスター覧

番号表示	別名	説明
$0	**$zero**	**常にゼロ**
$1	$at	アセンブラ用に予約
$2, $3	$v0, $v1	関数からの戻り値用
$4 ～ $7	$a0 ～ $a3	関数への引数用
$8 ～ $15	**$t0 ～ $t7**	**（主に）一時レジスタ**
$16 ～ $23	**$s0 ～ $s7**	**（主に）変数割り当て用**
$24, $25	**$t8, $t9**	**（主に）一時レジスタ**
$26, $27	$k0, $k1	OS用に予約
$28	$gp	グローバルポインタ
$29	$sp	スタックポインタ
$30	**$s8**	**（主に）変数割り当て用**
$31	$ra	リターンアドレス

MIPSのレジスタ-即値間演算命令

レジスタ-即値間の演算命令

(C言語)

```
y = x + 100;
```

ただし，変数 x, y の内容がそれぞれレジスタ s0, s1 に置かれているとする

(MIPSアセンブリ言語)

```
addu $s1, $s0, 100     # $s1 ← $s0 + 100
```

- 命令内で直接指定される定数を**即値**（immediate）と呼ぶ.
- 演算命令は，2つめの入力オペランドとして即値を取れる.
  - 1つめは必ずレジスタ．出力オペランドももちろんレジスタ

- 即値を取る addu 命令は，実際には addiu という命令として実行される（アセンブラが自動的に変換する．SPIMの実行画面にも注目）
- 即値を取る subu 命令は，実際には addiu 命令に符号反転した即値を渡すことで実行される.

例

(C言語)

```
y = x - 8;
z = 15;
```

ただし，変数 x, y, z の内容がそれぞれレジスタ s0, s1, s2 に置かれているとする

(MIPSアセンブリ言語)

```
subu $s1, $s0, 8          # $s1 ← $s0 - 8
or   $s2, $zero, 15       # $s2 ← 15
```

- 2行目のようにレジスタへの定数代入よく行われるので，li (load immediate) というマクロ命令でも書けるようになっている.

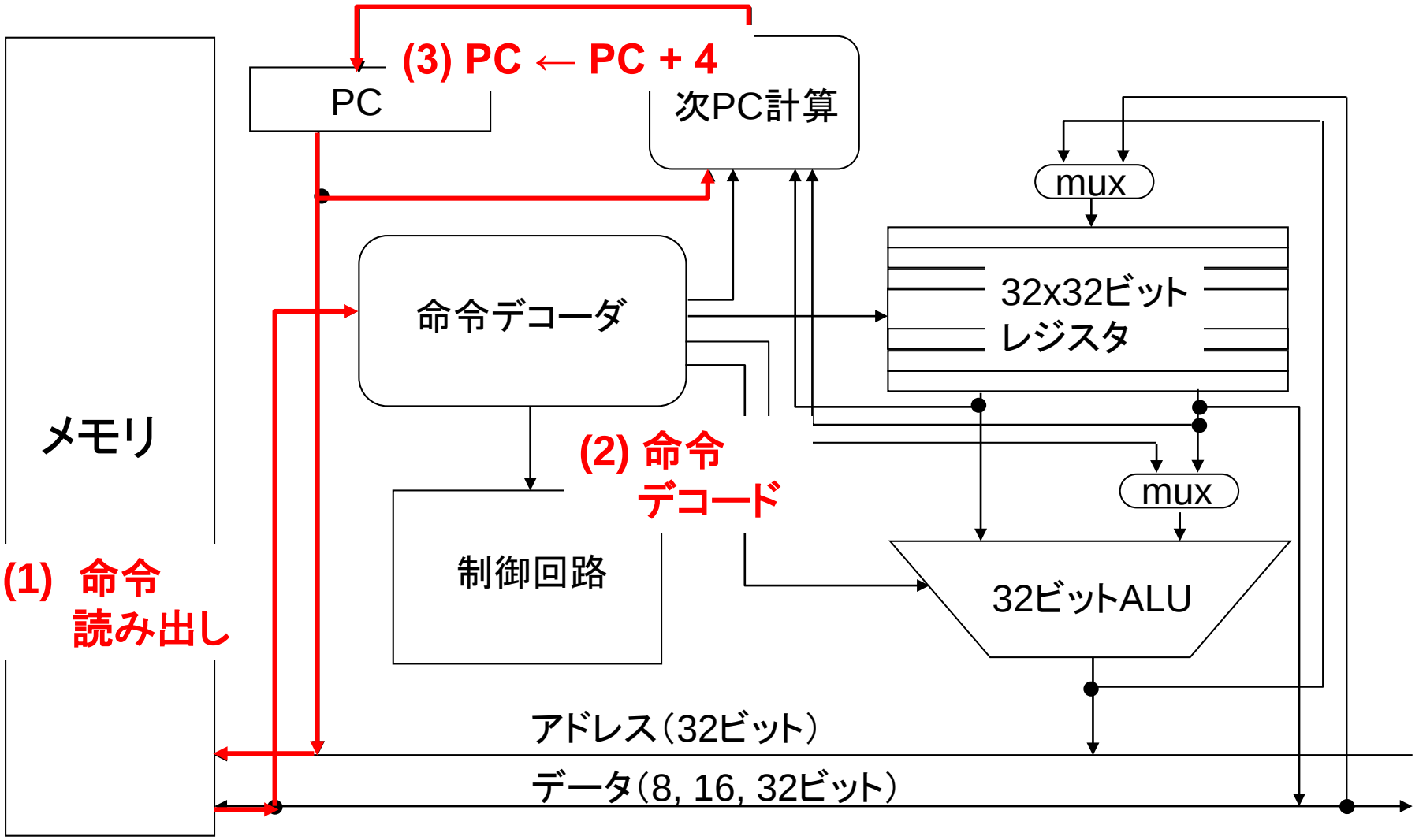
```
li       $s2, 15          # $s2 ← 15
```

資料: 主なマクロ命令

命令	説明
move $a, $b	$a ← $b　　# or $a, $zero, $b と等価
li $a, imm	$a ← imm　　# or $a, $zero, imm と等価 (load immediate)
nop	何もしない　　# sll $zero, $zero, $zero と等価 (no operation)

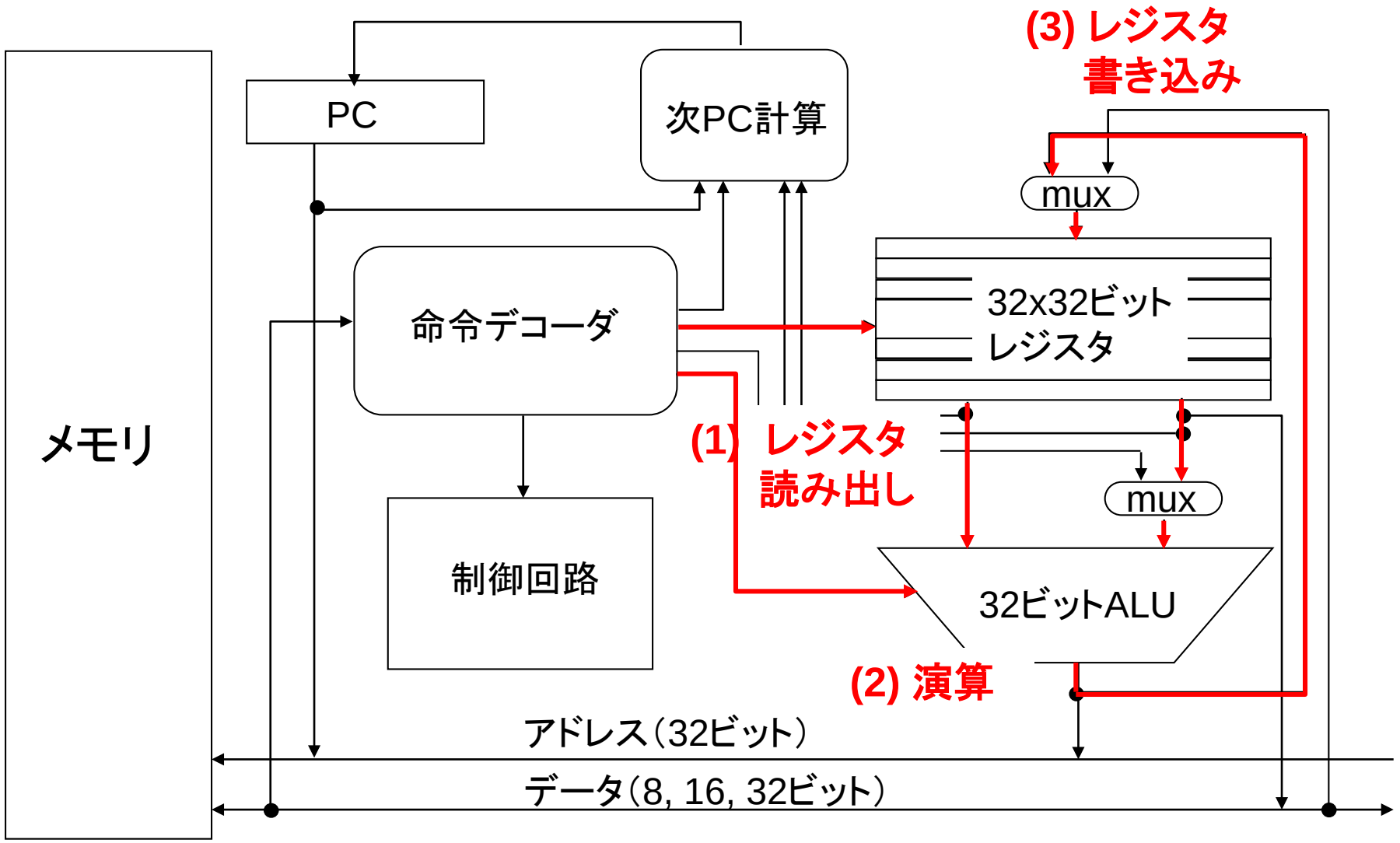
ハードウェアとして用意されているわけではなく，アセンブラが他の命令（の組み合わせ）に変換することによって実現される命令

命令フェッチと命令デコードの動作



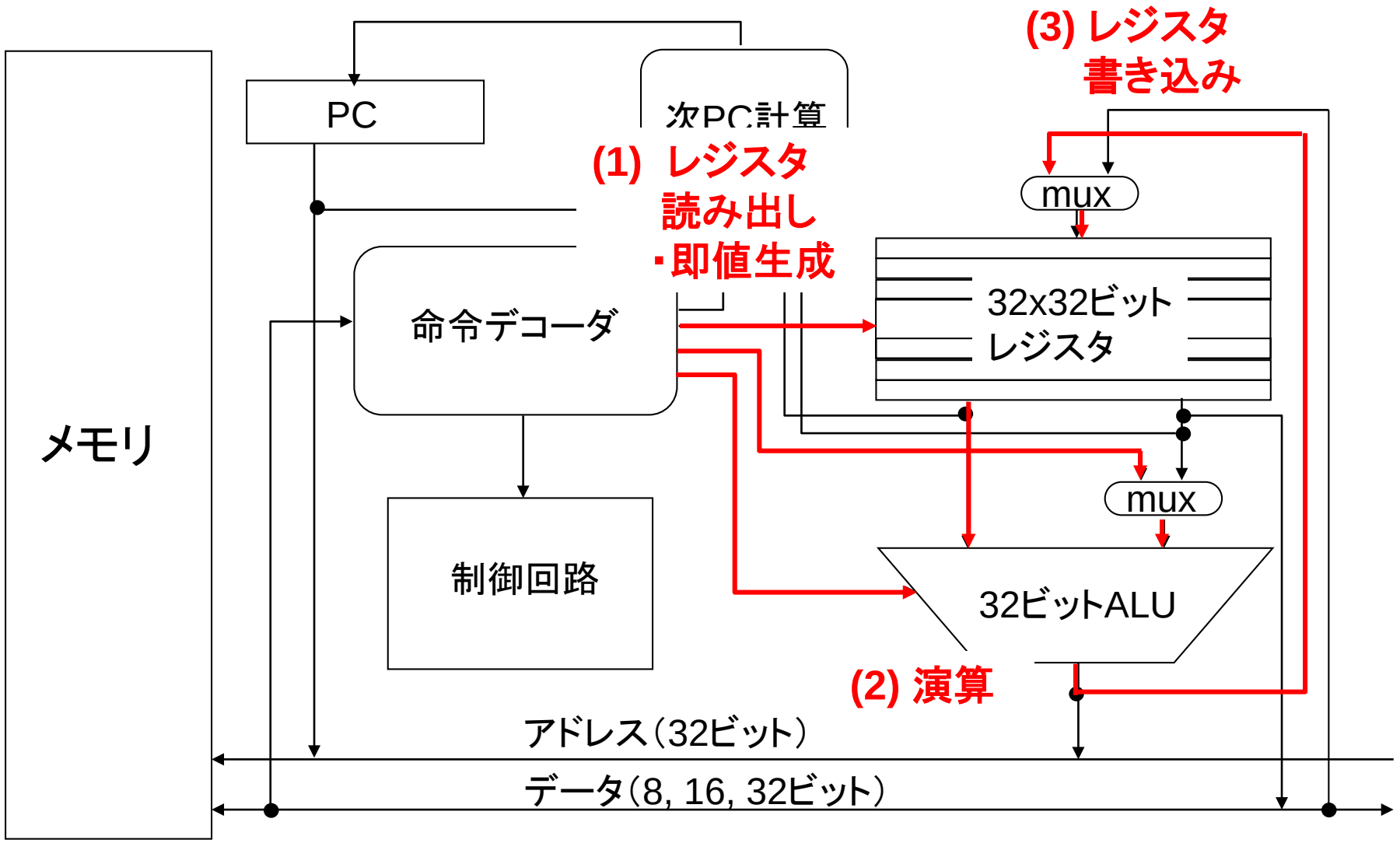
mux は選択回路

レジスタ間演算の動作



mux は選択回路

レジスタ-即値間演算の動作



mux は選択回路

MIPSのロード・ストア命令

ロード・ストア命令

- レジスタとメモリアドレスを指定して，相互間でデータ転送を行う
- メモリアドレスの指定方法をアドレッシングモードと呼ぶ
- MIPSの場合，アドレッシングモードは1つしかない

```
lw $s1, 12($s0)          # $s1 ← mem[12 + $s0]
```

- 操作対象のメモリアドレス（**実効アドレス**）はレジスタ s0 の値と定数 12 を加えたもの
- 例えばレジスタ s0 に 10000 が保存されていたとすると，アドレス 10012 から始まる 4 バイト (= 1 ワード) のデータを読み出し，レジスタ s1 に書き込む

資料: 主なロード・ストア命令

命令	説明
lw $t, offset($base)	$t ← mem[offset + $base] (load word)
sw $t, offset($base)	mem[offset + $base] ← $t (store word)

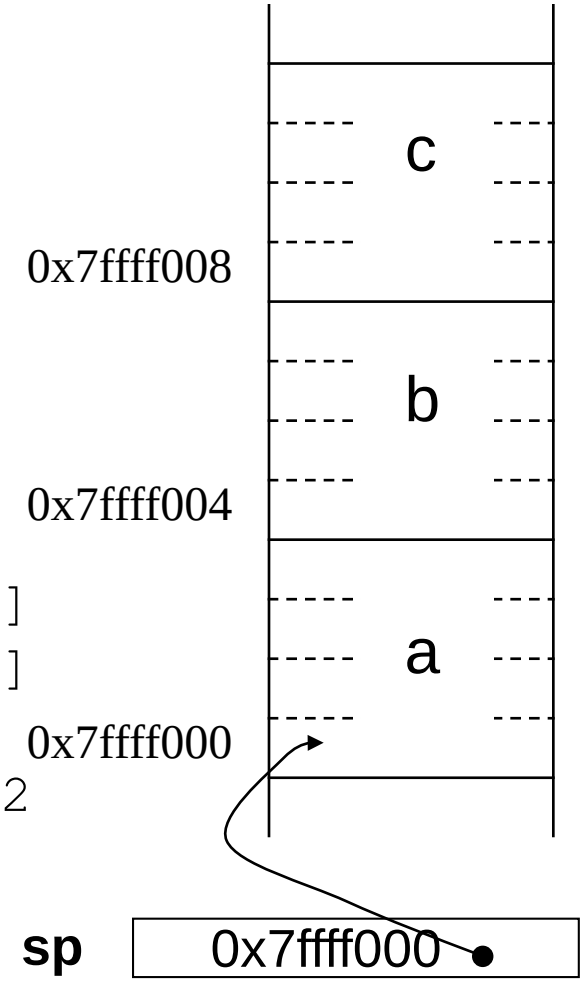
- 他に，half word 単位や byte 単位のロード命令，ストア命令がある

例

(C言語)

```
c = a + b;
```

ただし，各変数は右図のようにメモリに割り当てられているとし，レジスタ t0, t1, t2が自由に使えるとする．以下同様．



(MIPSアセンブリ言語)

```
lw $t0, 0($sp)      # $t0 ← mem[$sp + 0]
lw $t1, 4($sp)      # $t1 ← mem[$sp + 4]
addu $t2, $t0, $t1  # $t2 ← $t0 + $t1
sw $t2, 8($sp)      # mem[$sp + 8] ← $t2
```

※ 実際のアドレスがいくつになるかは場合により異なる

再掲: レジスター覧

番号表示	別名	説明
$0	**$zero**	**常にゼロ**
$1	$at	アセンブラ用に予約
$2, $3	$v0, $v1	関数からの戻り値用
$4 ～ $7	$a0 ～ $a3	関数への引数用
$8 ～ $15	**$t0 ～ $t7**	**(主に)一時レジスタ**
$16 ～ $23	**$s0 ～ $s7**	**(主に)変数割り当て用**
$24, $25	**$t8, $t9**	**(主に)一時レジスタ**
$26, $27	$k0, $k1	OS用に予約
$28	$gp	グローバルポインタ
$29	**$sp**	**スタックポインタ**
$30	**$s8**	**(主に)変数割り当て用**
$31	$ra	リターンアドレス

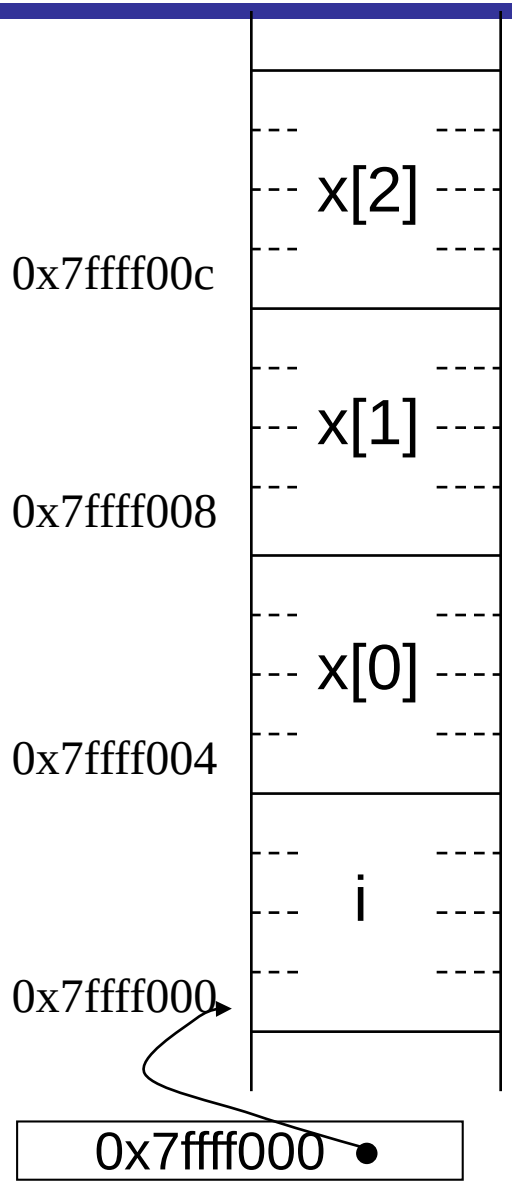
例

(C言語)

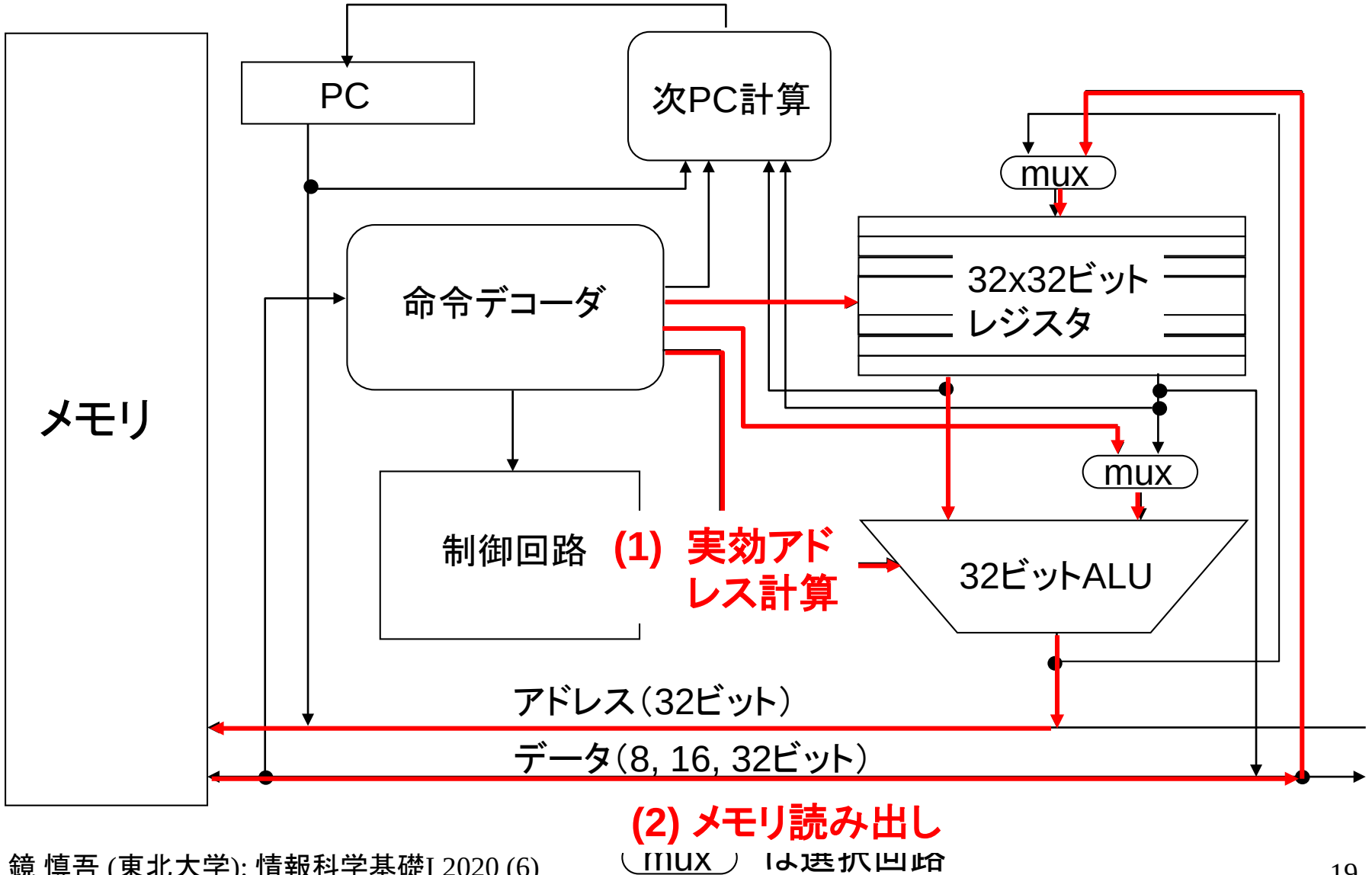
```
int i;
int x[3];
/* 中略 */
x[i] = 300;
```

(MIPSアセンブリ言語)

```
addu $t0, $sp, 4    # $t0 ← $sp + 4
lw   $t1, 0($sp)    # $t1 ← mem[$sp + 0]
sll  $t1, $t1, 2    # $t1 ← $t1 × 4
addu $t0, $t0, $t1  # $t0 ← $t0 + $t1
li   $t2, 300       # $t2 ← 300
sw   $t2, 0($t0)    # mem[$t0] ← $t2
```

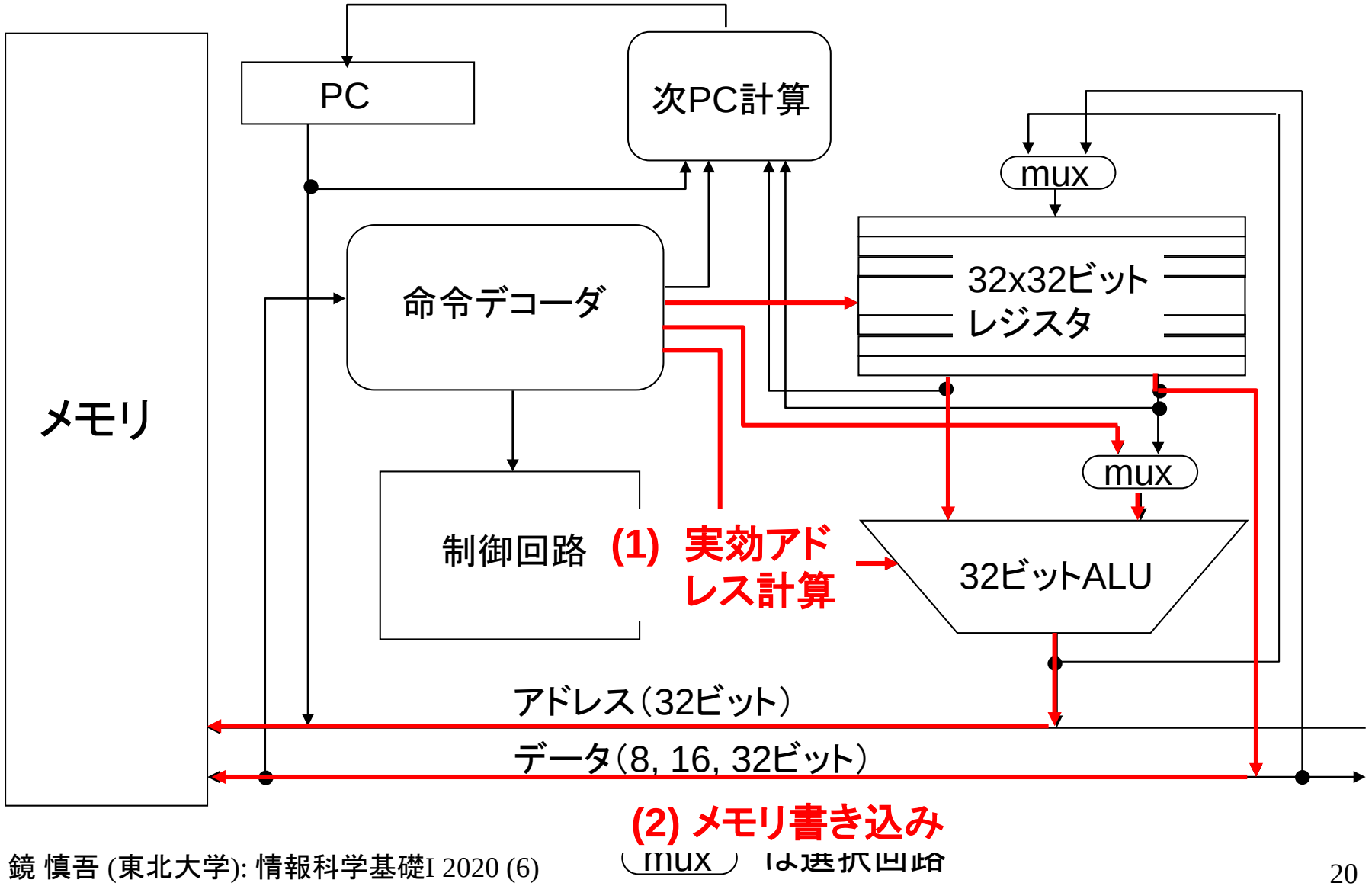


ロード命令の動作



mux は選択回路

ストア命令の動作


メモリ
PC
次PC計算
mux
命令デコーダ
32x32ビット
レジスタ
mux
制御回路
(1) 実効アドレス計算
32ビットALU
アドレス（32ビット）
データ（8, 16, 32ビット）
(2) メモリ書き込み
mux は選択回路

例題（余談）: スーパーマリオブラザーズ256W

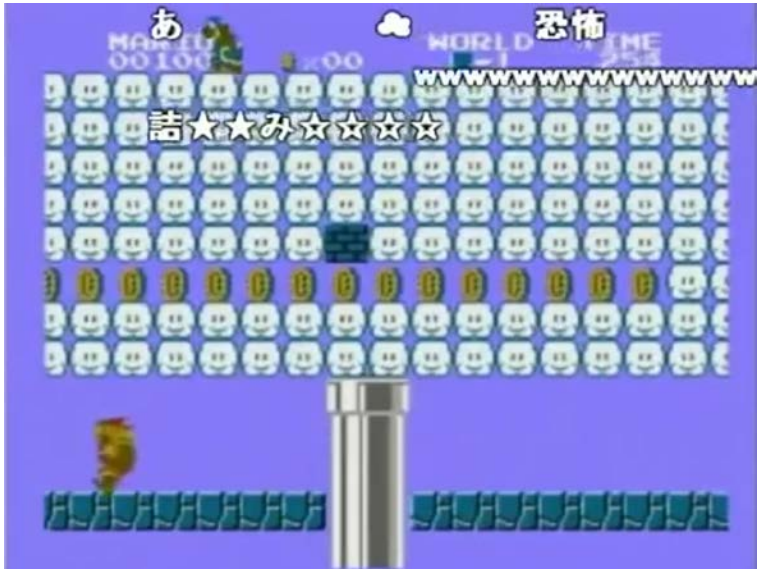
ファミリーコンピュータ用ゲーム「スーパーマリオブラザーズ」（任天堂, 1985）では，以下のような操作により，通常は 1 ～ 8 までしか存在しない「ワールド」（ゲーム内のステージの呼称）が，テニスのプレイヤの位置に応じて最大 256 種類出現する現象が生じた.

- ゲームプレイ中に電源を切らずにゲームカセットをゲーム機本体から引き抜き，
- 別のゲーム「テニス」（任天堂, 1984）のカセットを挿入してリセットボタンを押し，しばらくプレイした後に，
- 同様にカセットを抜き，スーパーマリオブラザーズに戻してリセットボタンを押し，Aボタンを押しながらプレイを開始する.

この事実から推測できることを述べよ.

（2014年度期末試験・改題）

https://www.youtube.com/watch?v=AXV05nwZtxo

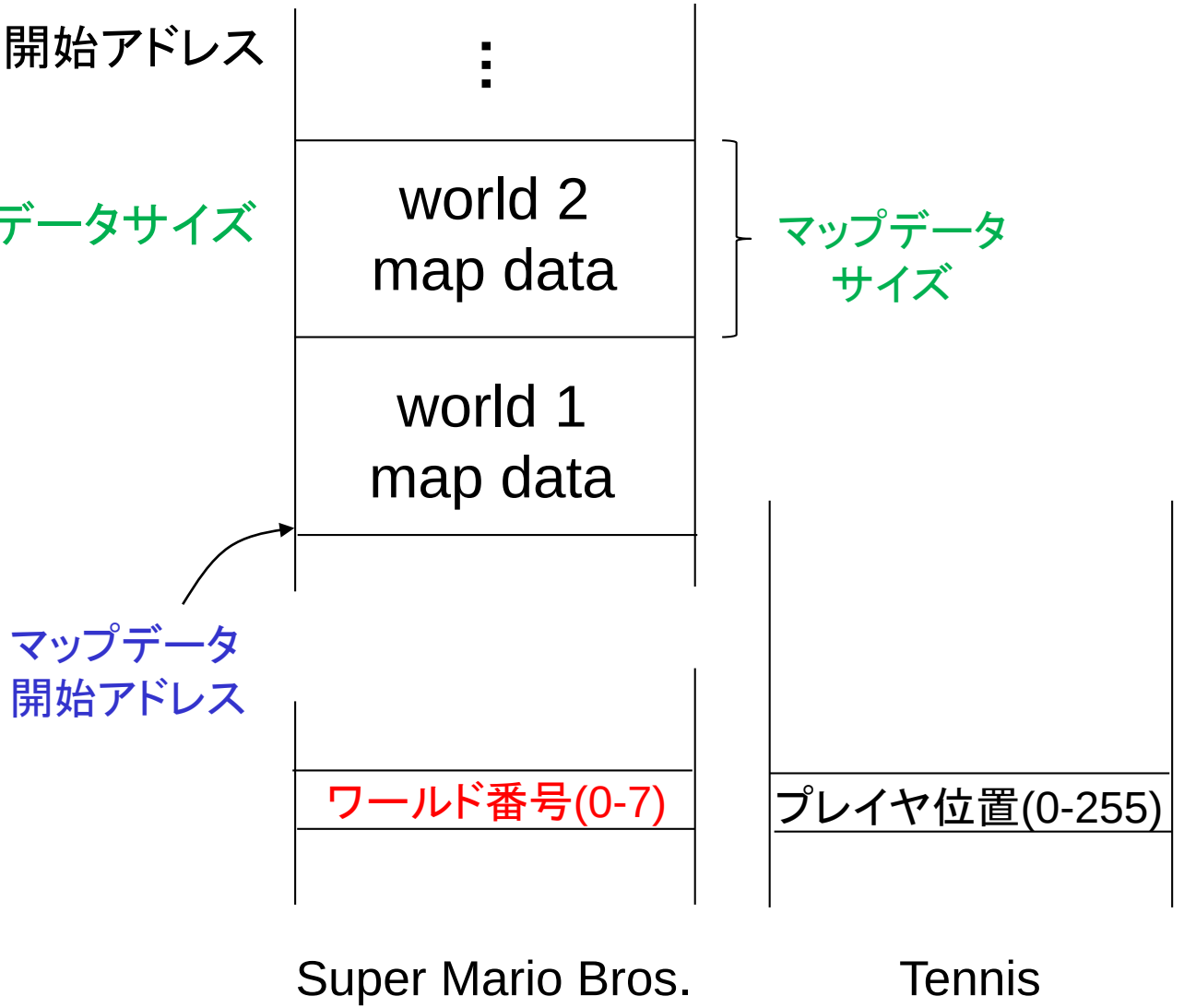


https://www.youtube.com/watch?v=AXV05nwZtxo

各ワールドのマップデータ開始アドレス

= マップデータ開始アドレス

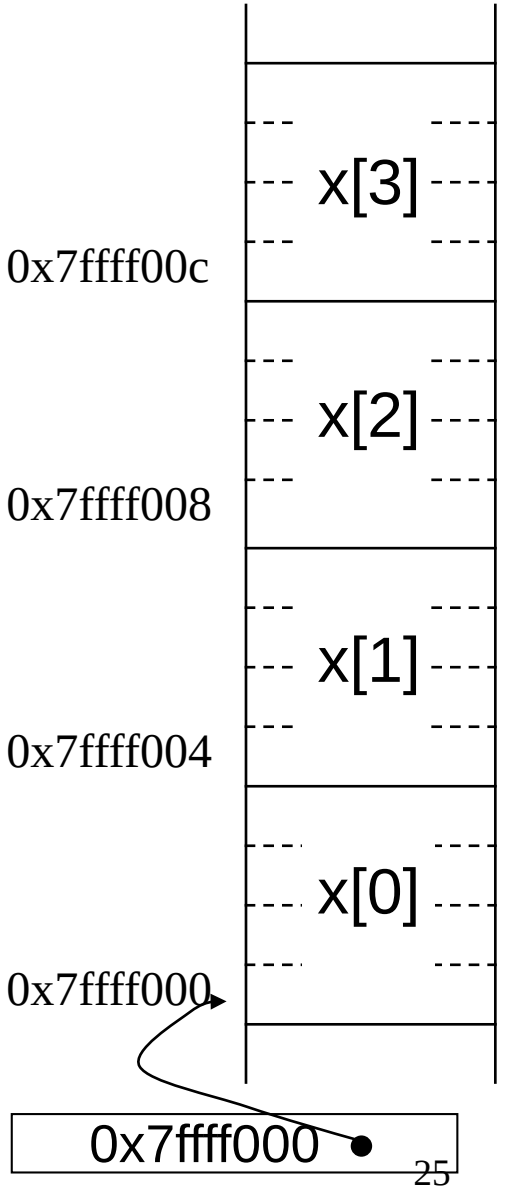
+ ワールド番号 × マップデータサイズ



練習問題

4要素の4バイト整数型からなる配列が右図のようにメモリに配置されているとする. x[0] … x[3] に格納されている値がそれぞれ 1, 3, 5, 7 の状態から, 以下のコードを実行した. 実行後の x[0] … x[3] の値はどうなっているか.

```
lw $t0, 0($sp)
addu $t0, $t0, 3
sw $t0, 0($sp)
lw $t1, 4($sp)
lw $t0, 8($sp)
sll $t0, $t0, $t1
sw $t0, 8($sp)
lw $t0, 12($sp)
slt $t0, $t0, $t1
sw $t0, 12($sp)
```



解答例

```
lw $t0, 0($sp)          # $t0 ← x[0] (= 1)
addu $t0, $t0, 3        # $t0 ← $t0 + 3 (= 1 + 3)
sw $t0, 0($sp)          # x[0] ← $t0 (= 4)
lw $t1, 4($sp)          # $t1 ← x[1] (= 3)
lw $t0, 8($sp)          # $t0 ← x[2] (= 5)
sll $t0, $t0, $t1       # $t0 ← $t0 << $t1 (= 5 << 3)
sw $t0, 8($sp)          # x[2] ← $t0 (= 40)
lw $t0, 12($sp)         # $t0 ← x[3] (= 7)
slt $t0, $t0, $t1       # $t0 ← ($t0 < $t1) ? 1 : 0
sw $t0, 12($sp)         # x[3] ← $t0 (= 0)
```

結局 x[0] … x[3] は 4, 3, 40, 0 になる.