#fis1

東北大学 工学部 機械知能・航空工学科
2019年度 クラス C D

情報科学基礎 I

10. 組合せ回路
(教科書3.4〜3.5節)

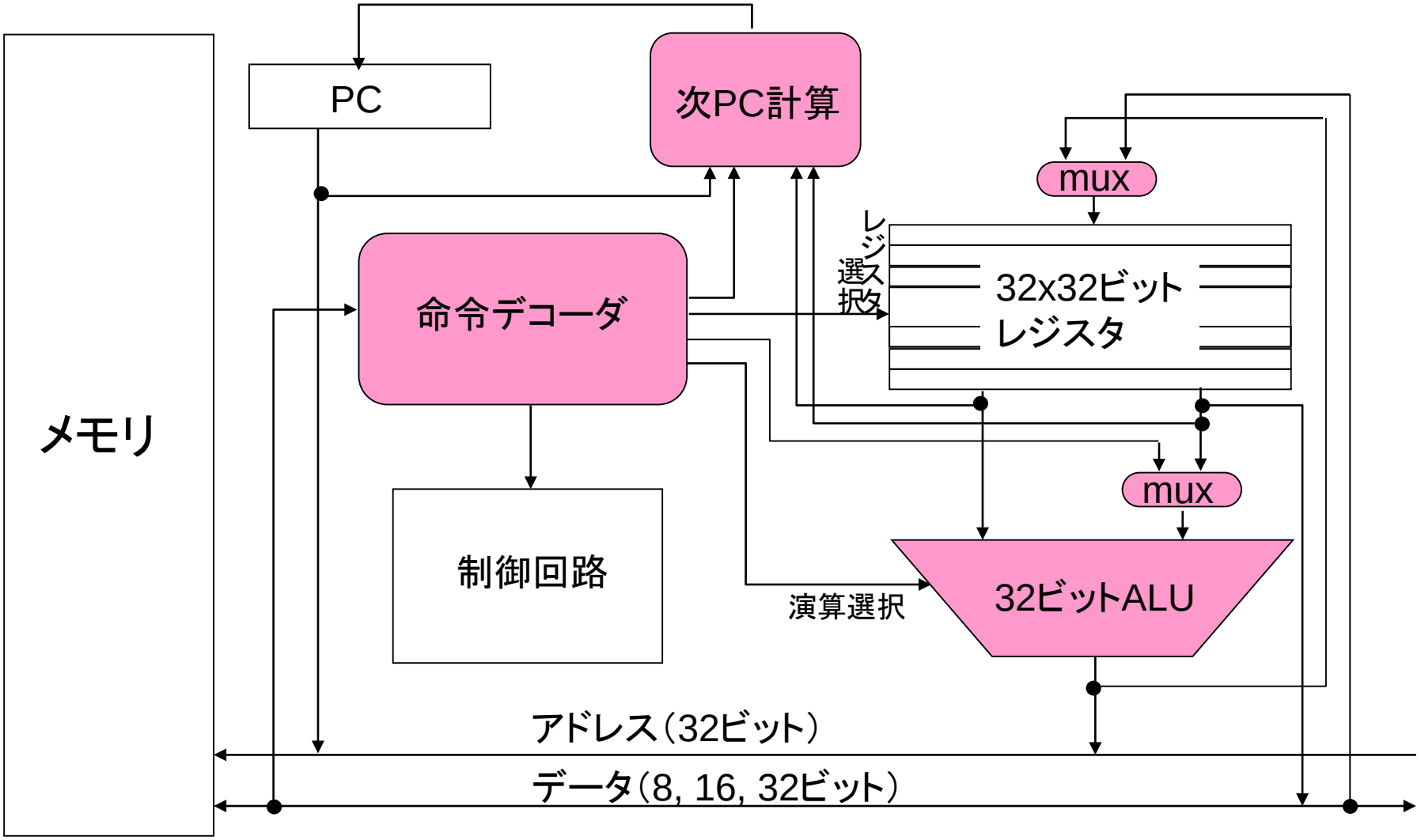
大学院情報科学研究科
鏡 慎吾
http://www.ic.is.tohoku.ac.jp/~swk/lecture/

組合せ論理回路

$$y_i = f_i(x_1, x_2, \ldots, x_n),\ i = 1, 2, \ldots, m$$

- ある時点での出力が，その時点の入力のみで決まる（記憶を持たない）回路
  - フィードバックが存在しない（入力→出力の方向にだけゲートが接続されている）
- 原理的には，n 入力の論理関数が m 個並んでいるものだと考えればよい

復習: MIPSの構造



色つきの部分が組合せ回路

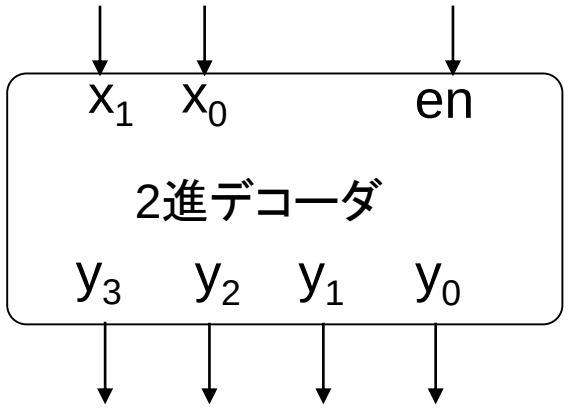
組合せ論理回路の構成方法

- 原理上は，必ず積和形回路で表すことができる，しかし
  - nが大きい場合，簡単化の計算に膨大なコストがかかる
  - それが最適とは限らない
- 算術論理演算のように入出力関係の規則性が高い場合は，その規則性に注目して回路を組み立てる方がよい

- 複数の回路を接続するための部品
  - **2進デコーダ**，**マルチプレクサ**
- 複数回路の接続例
  - ALU，汎用レジスタ群
- 各種演算回路
  - **加算器**と算術演算，論理演算

2進デコーダ

複数の出力信号のうち
（高々）1 本を選んで 1 にする



- en = 0 のとき: 全出力を0とする
- en = 1 のとき: 入力を2進数 k と見なして，出力 y_k を1，他を0とする

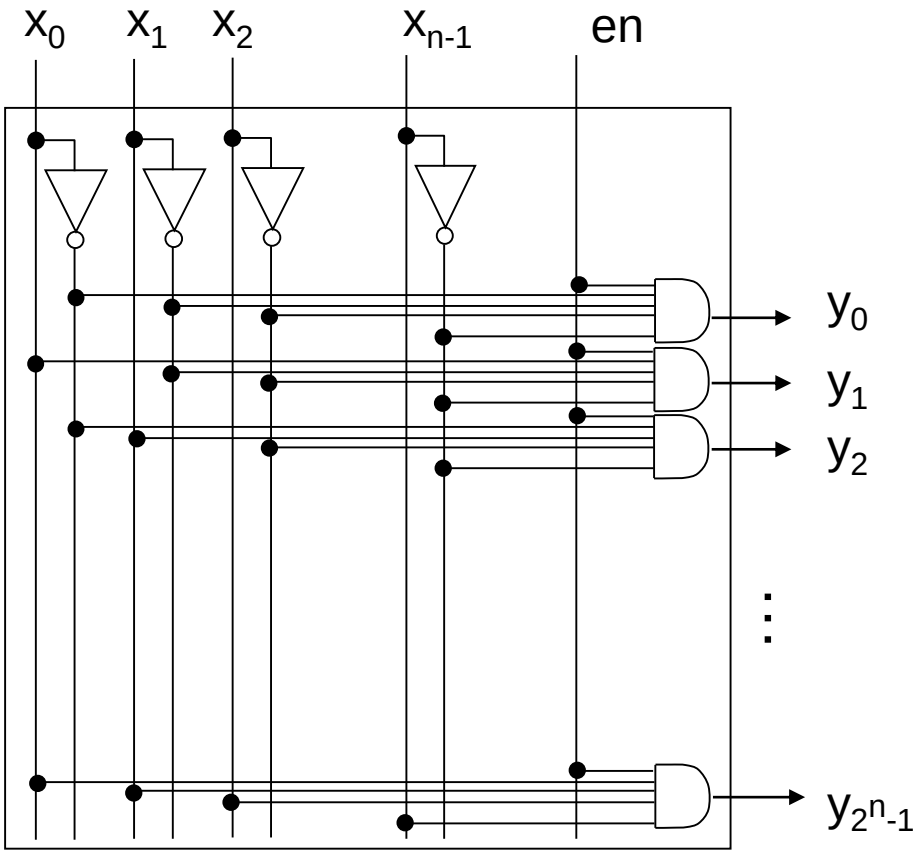
例: $x_1 = 1, x_0 = 0$ のとき，入力は2進数で「2」を表すので，y_2 のみが1 となる

- エンコード: 一般に，注目している量に適当な数値（符号）を与えること
- デコード: エンコードの逆
- この例では，「何番目の信号線か?」を2進数として符号化している
- en は enable の略で，活性化信号などと訳される

2進デコーダの真理値表と回路図

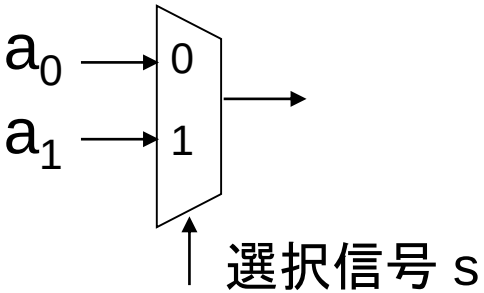
- 各 y_i について真理値表を書くと，1行だけ出力が1になるような表となる
- 簡単化は全くできないので，主加法標準形のまま回路化

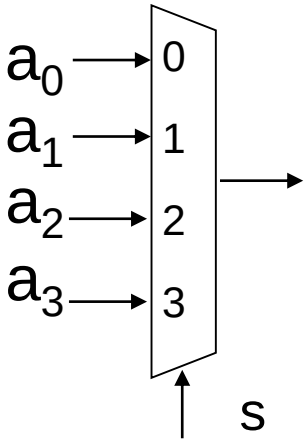
x_2	x_1	x_0	$y_7y_6\cdots y_1y_0$
0	0	0	0000 0001
0	0	1	0000 0010
0	1	0	0000 0100
0	1	1	0000 1000
1	0	0	0001 0000
1	0	1	0010 0000
1	1	0	0100 0000
1	1	1	1000 0000



マルチプレクサ（セレクタ）

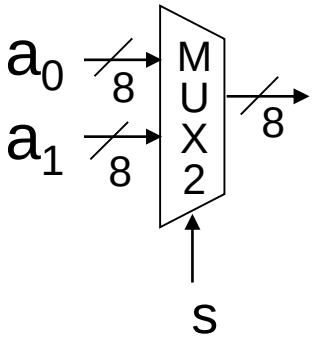
複数の入力信号のうち 1 本を出力側に通す選択回路





s = i なら a_i を選ぶ

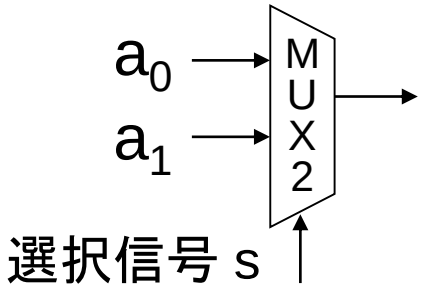
多ビットをまとめて選択するものも同様の記号で表す



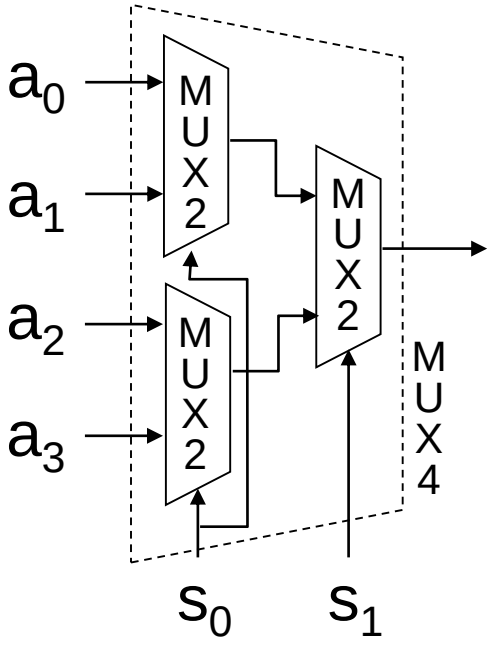
- 短い斜線と数字は，複数ビットをまとめたことを表示している
  （自明な場合，興味のない場合は適宜省略）
- 記号の形状は，台形だったり楕円だったりといろいろな流儀がある

マルチプレクサの真理値表と構成

2入力

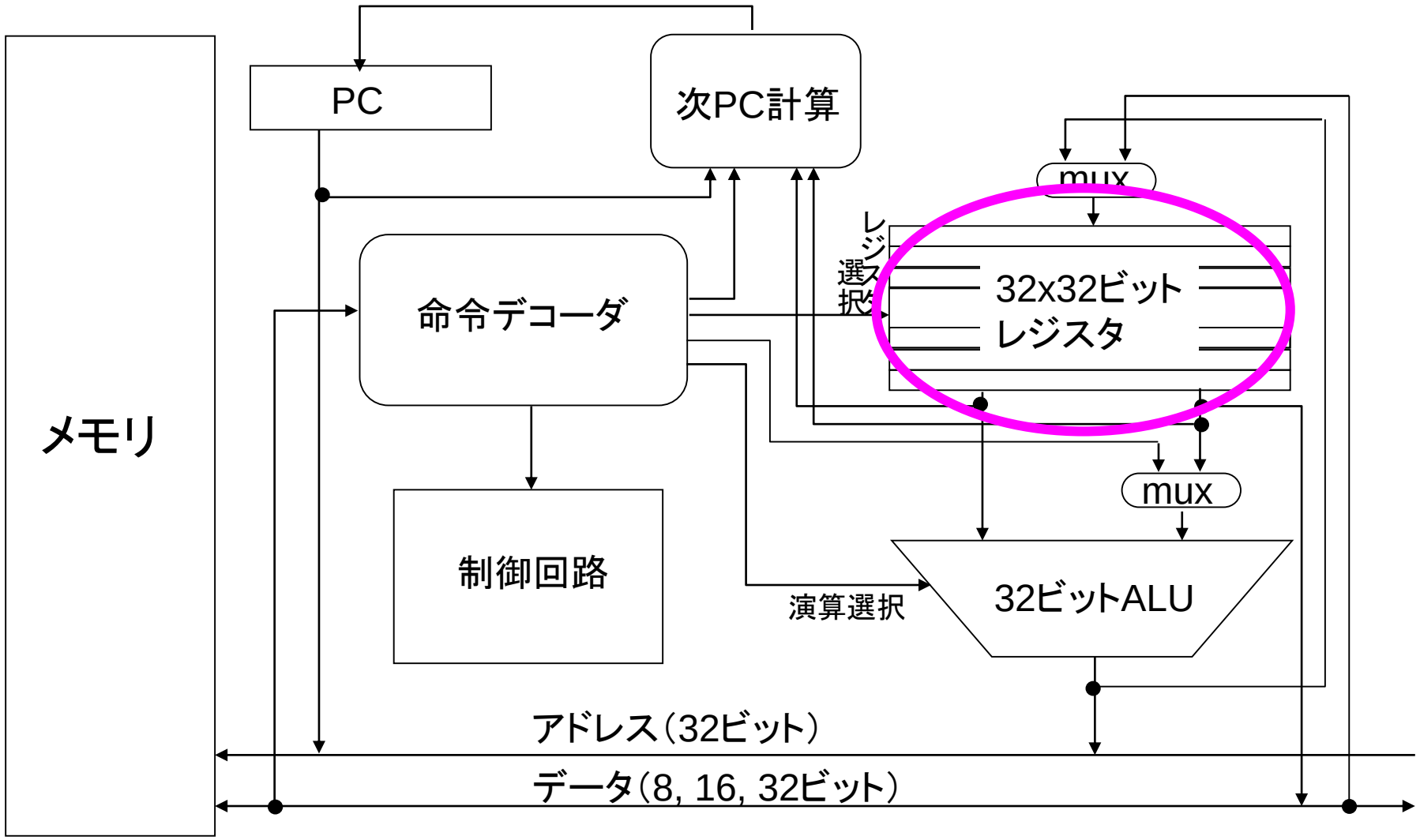


4入力

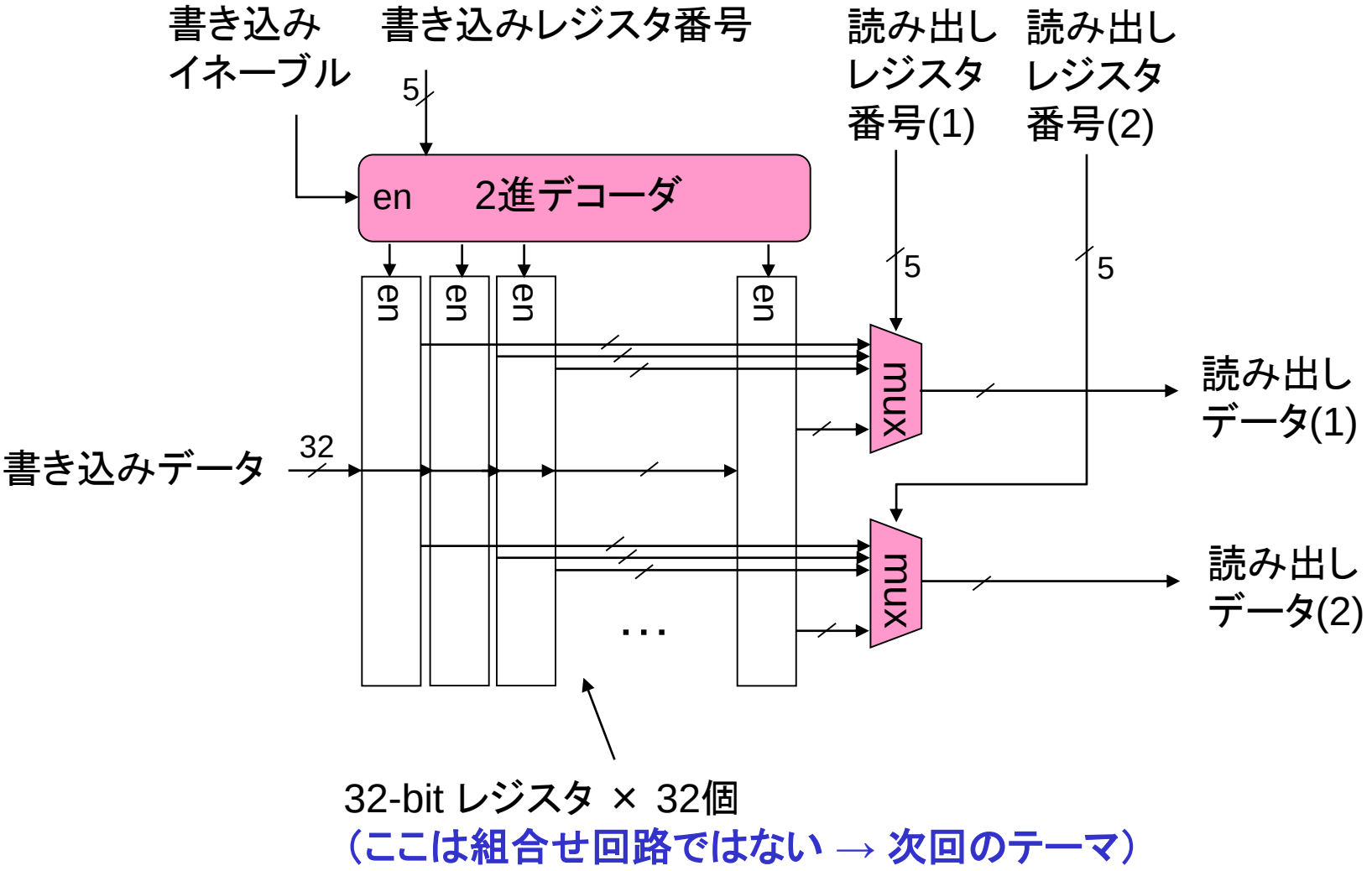


a_0	a_1	s	mux2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

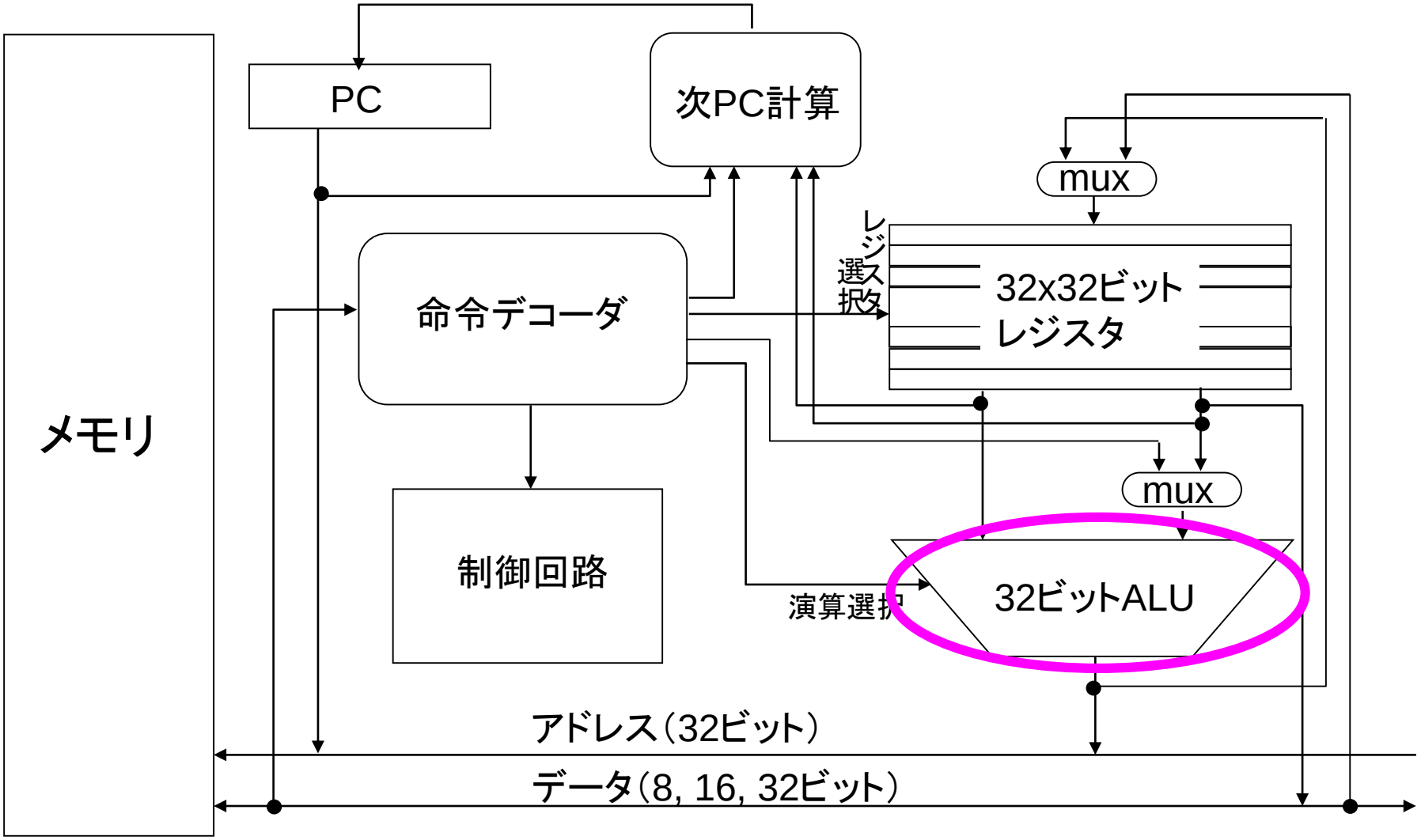
復習: MIPSの構造


メモリ
PC
次PC計算
命令デコーダ
制御回路
mux
レジスタ選択
32x32ビット
レジスタ
mux
演算選択
32ビットALU
アドレス(32ビット)
データ(8, 16, 32ビット)

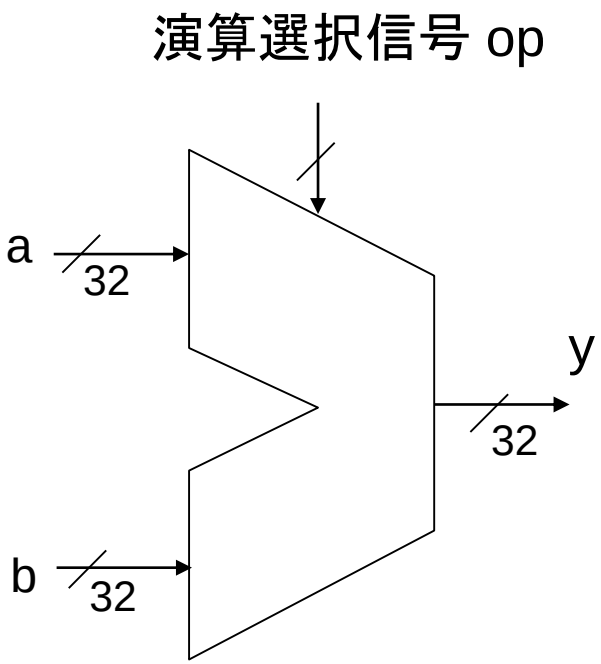
32×32ビットレジスタ (1入力2出力)

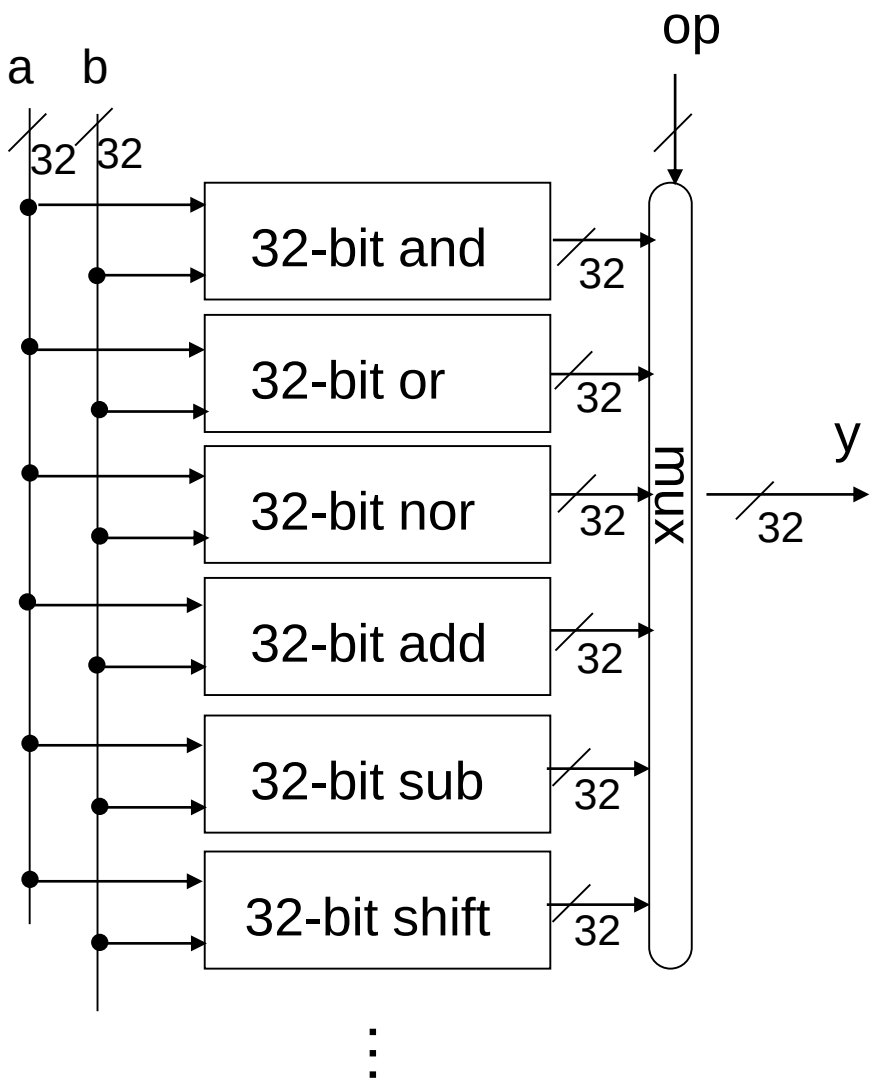


復習: MIPSの構造

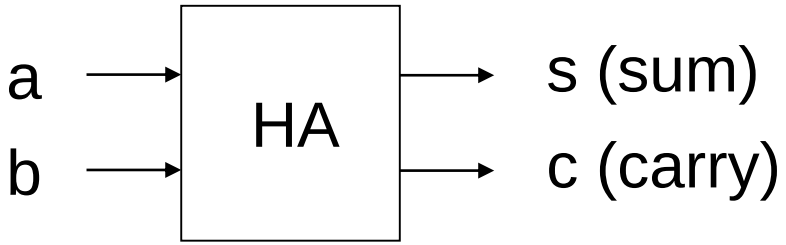

メモリ
PC
次PC計算
mux
レジスタ選択
32x32ビット
レジスタ
命令デコーダ
制御回路
mux
32ビットALU
演算選択
アドレス(32ビット)
データ(8, 16, 32ビット)

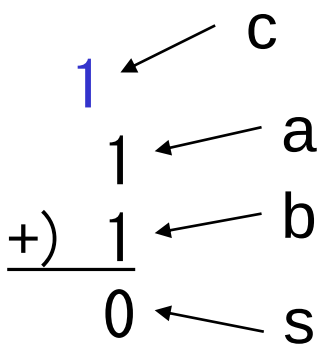
#fis1 ALU (Arithmetic and Logic Unit) の構成例





半加算器 (half adder)

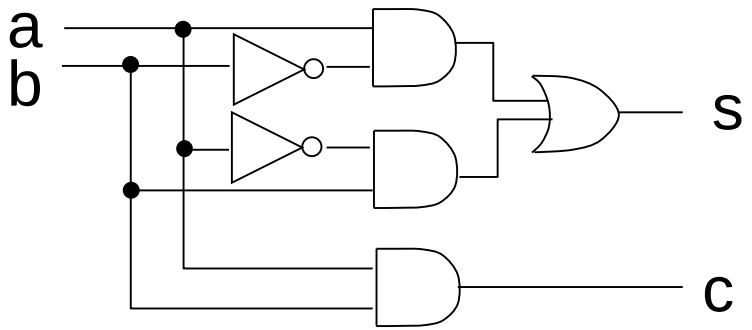




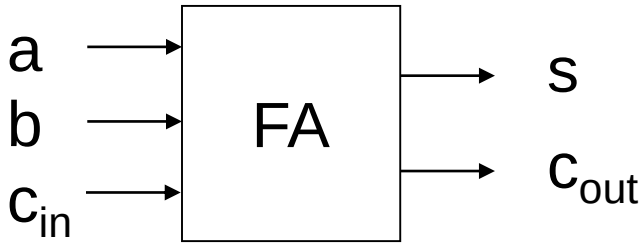
a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

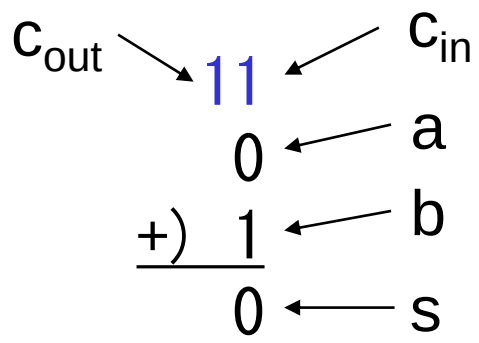
$$s = \bar{a}b + a\bar{b} \ (= a \oplus b)$$

$$c = ab$$

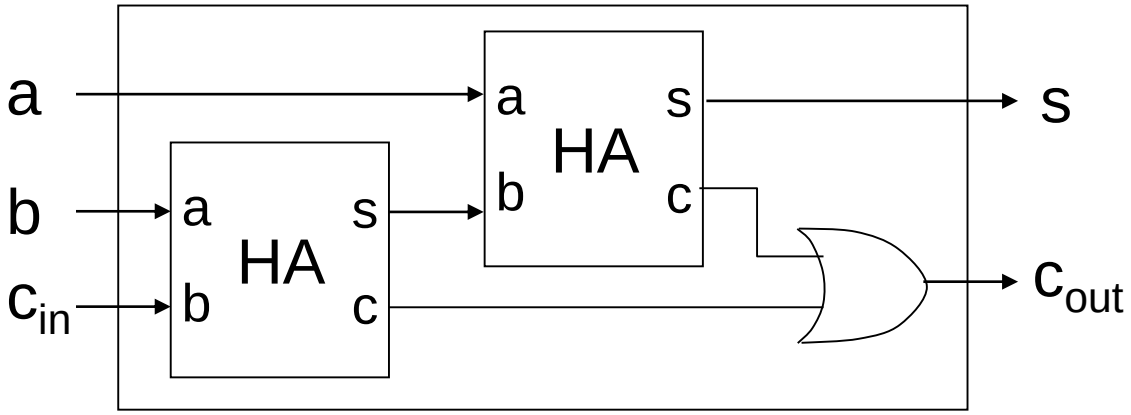


全加算器 (full adder)

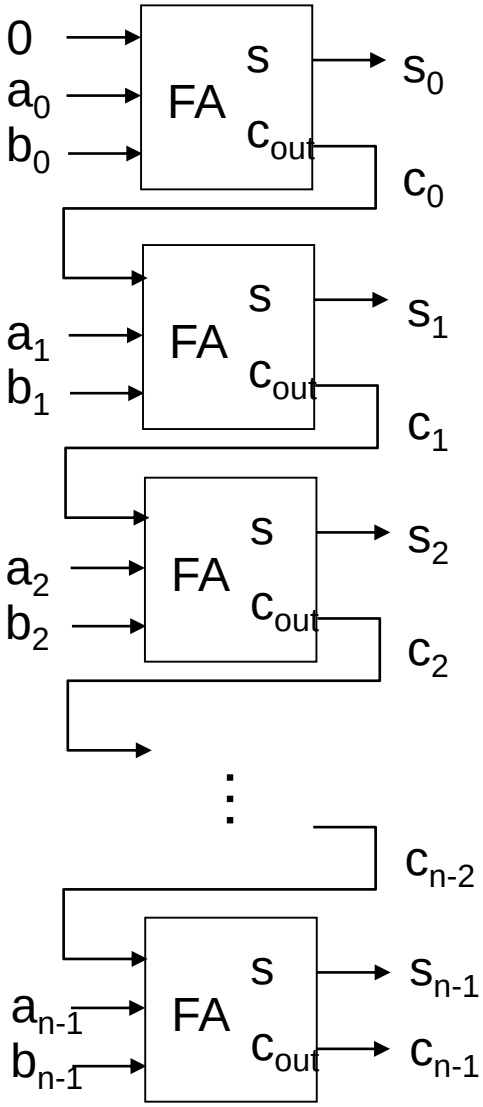




前の位からの繰り上がりを考慮する．半加算器が2つ必要



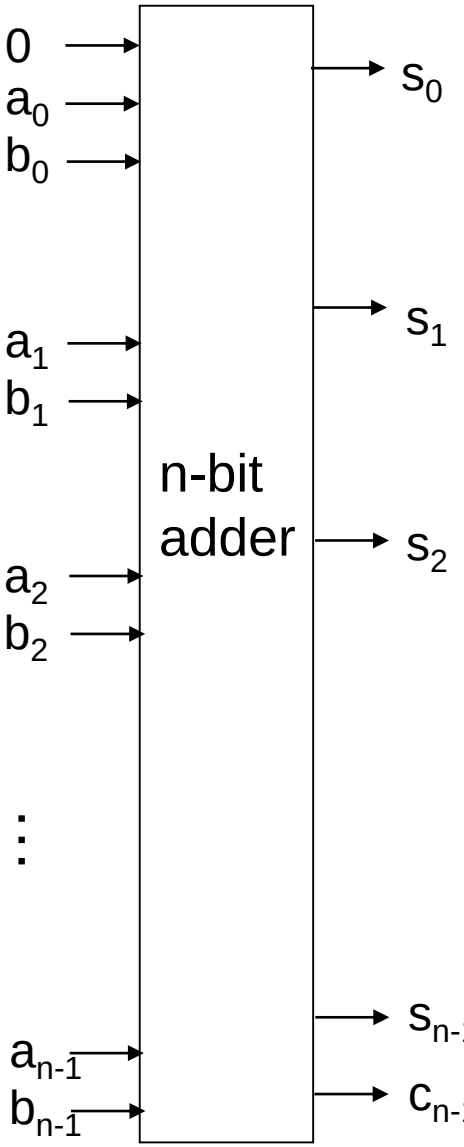
n-ビット加算器



リプルキャリー型加算器と呼ばれる

- nに比例して遅延が蓄積するため，決して速い回路ではない
- より高速な（しかし回路規模の大きい）加算回路も広く用いられている（e.g. キャリー先読み型加算器）

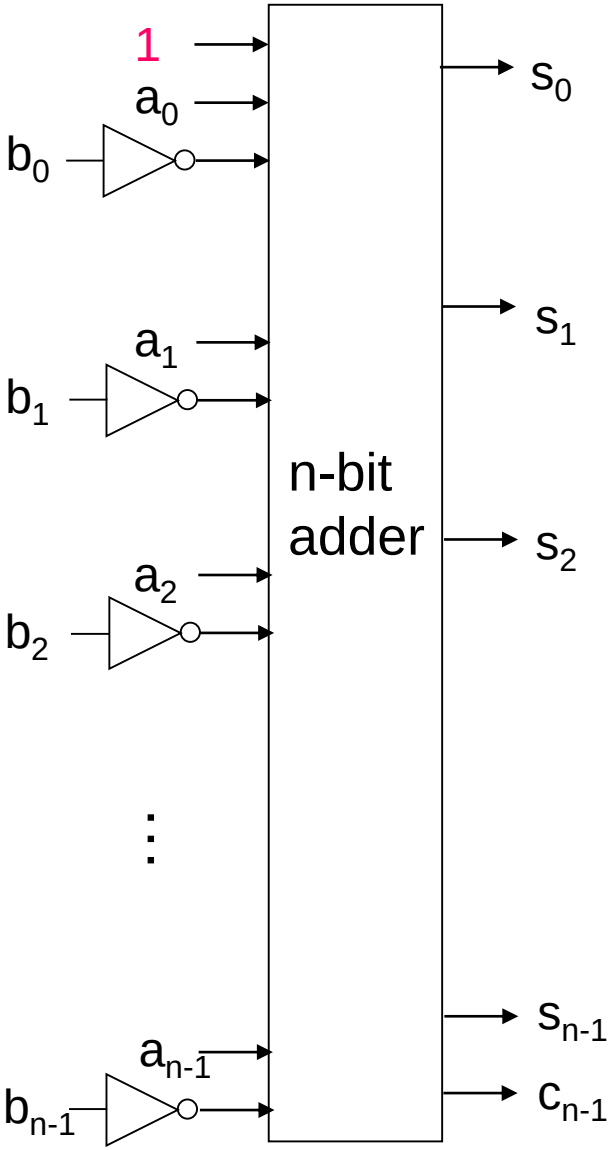
n-ビット減算器



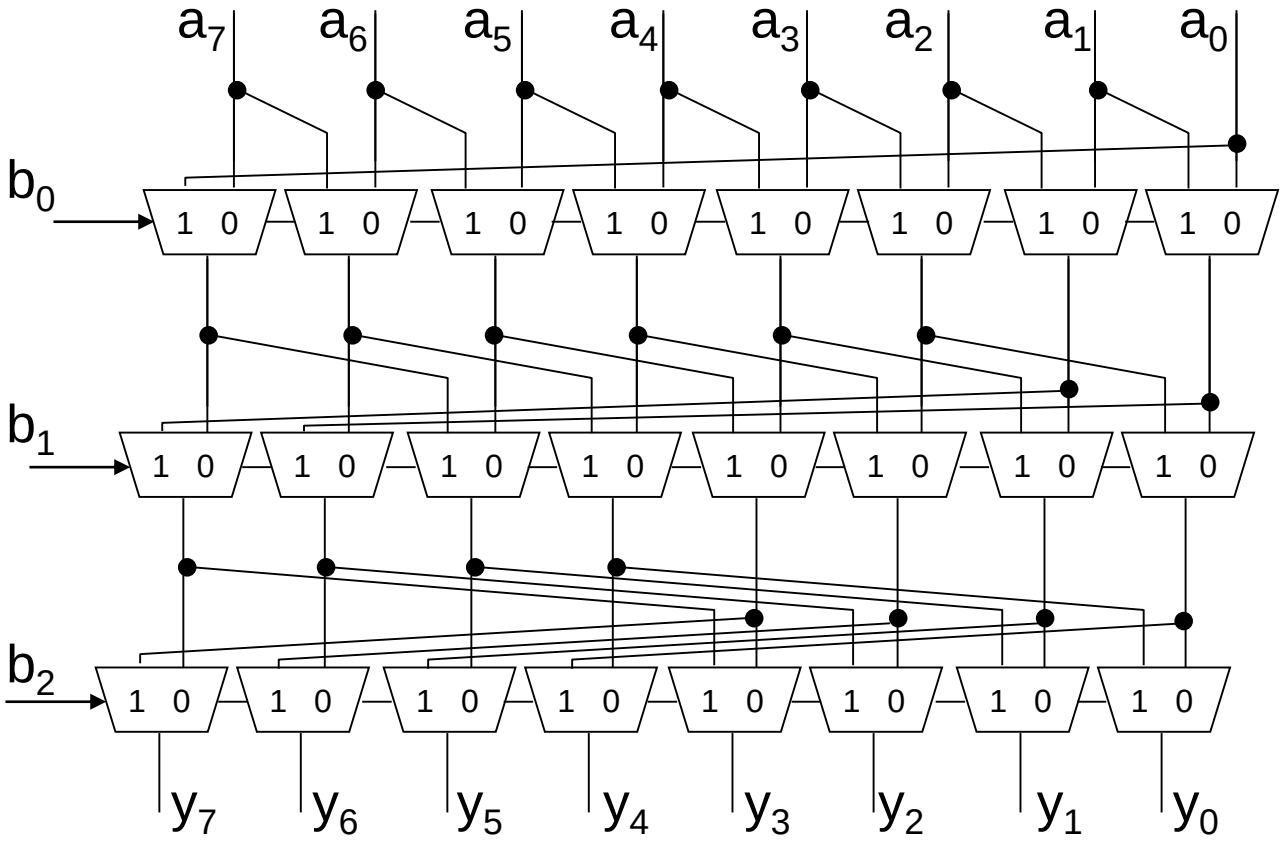
$$a - b = a + (-b)$$

入力を変えるだけで減算器になる

(よって p.12 のように加算器と減算器を独立して用意する必要は普通はない)



バレルシフタ(ローテータ)



1ビット右循環シフト

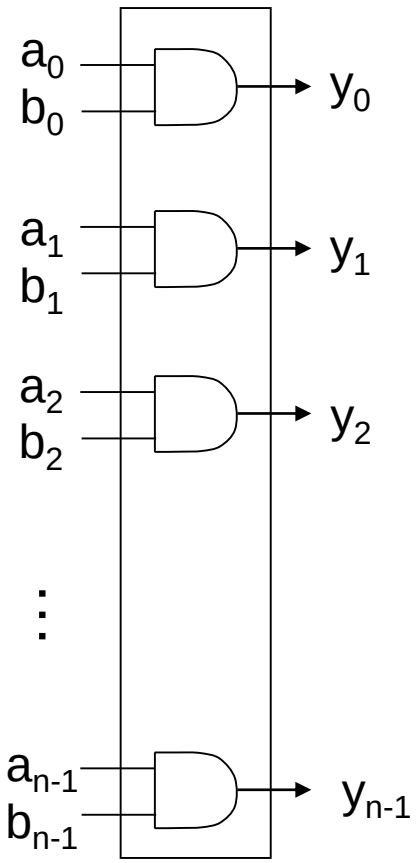
2ビット右循環シフト

4ビット右循環シフト

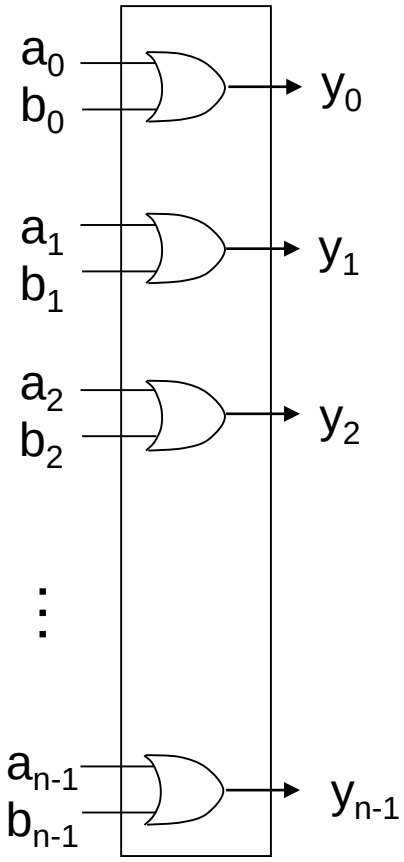
N ビット値 a を b ビット右循環シフトしたものを y として出力する組合せ回路

- b ビット左循環シフトは，N – b ビット右循環シフトと等価
- b ビット左シフトは，左循環シフト出力のLSB側 b ビットを 0 にする
- b ビット右シフトは，右循環シフト出力のMSB側 b ビットを 0 にする

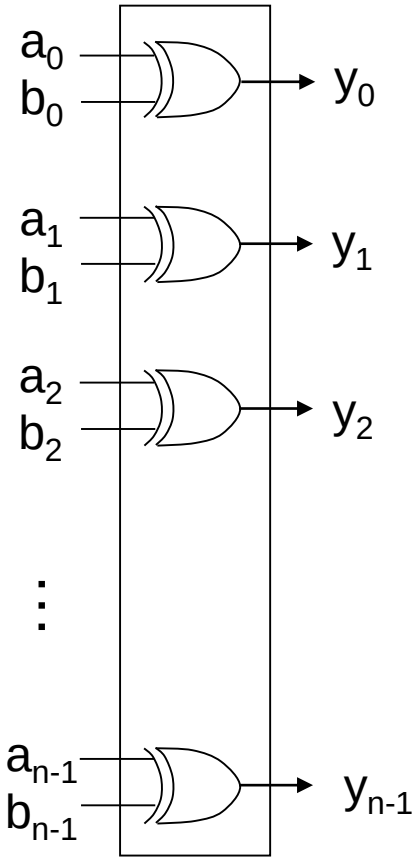
ビットごと論理演算器



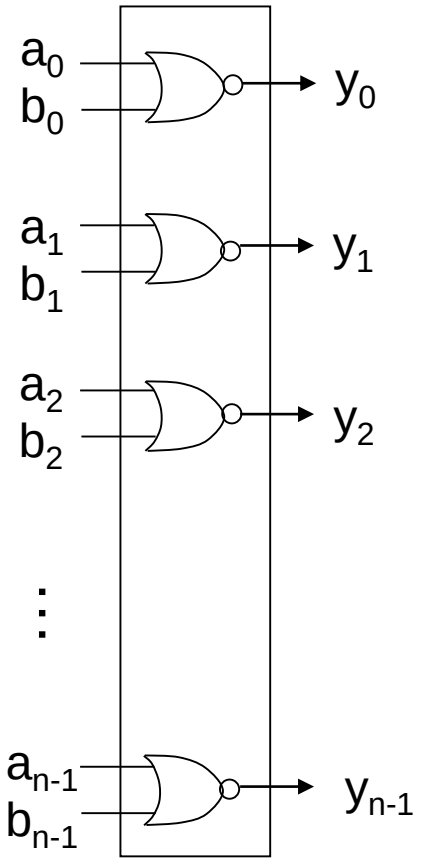
y = a & b



y = a | b

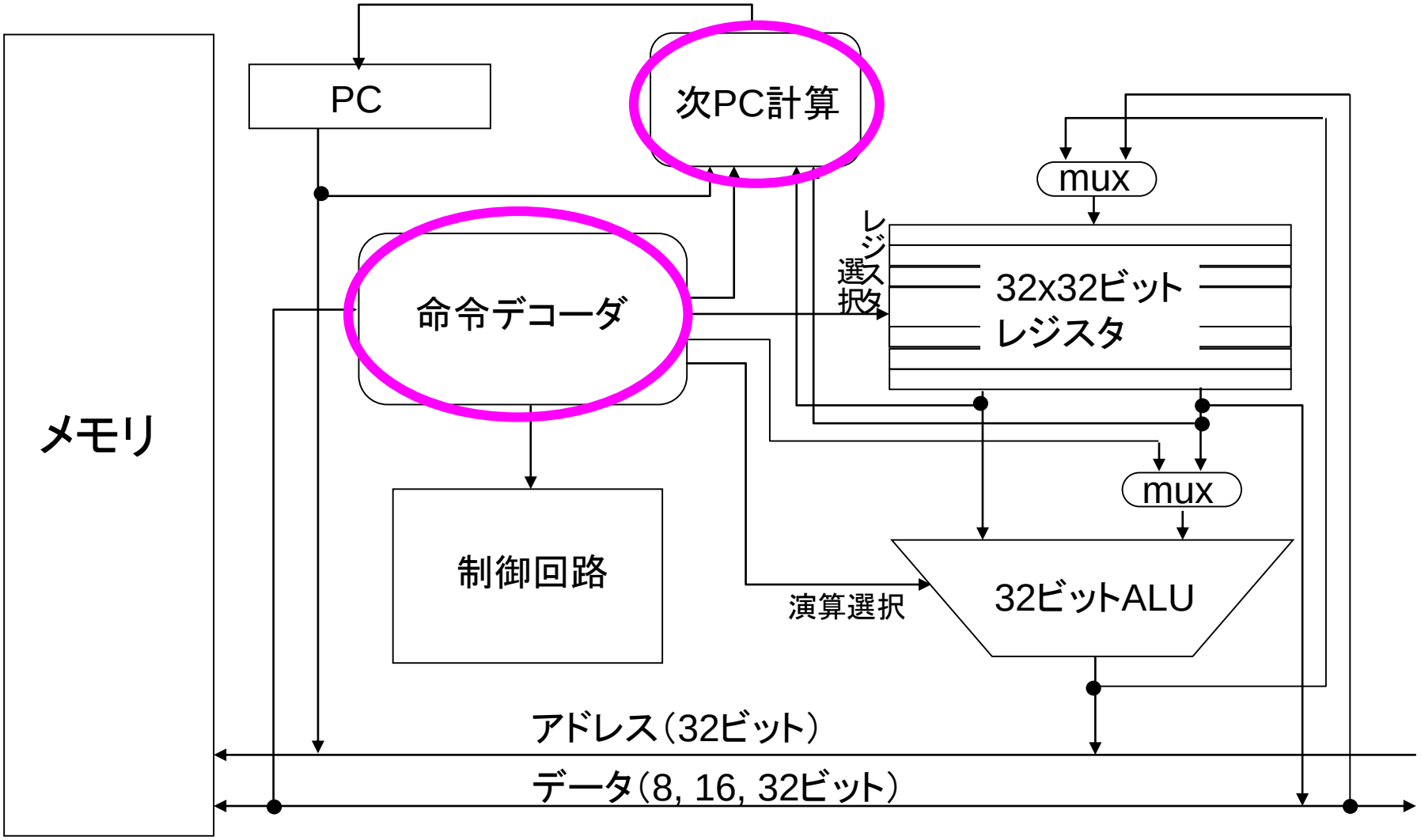


y = a ^ b



y = ~(a | b)

復習: MIPSの構造


メモリ
PC
次PC計算
命令デコーダ
制御回路
mux
レジスタ選択
32x32ビット
レジスタ
mux
演算選択
32ビットALU
アドレス(32ビット)
データ(8, 16, 32ビット)

参考: 命令デコーダと分岐ユニット

- **命令デコーダ**は，32ビットの命令を入力として，命令の解釈結果を出力する組合せ回路である．出力信号は例えば:
  - 命令種別（レジスタ演算，即値演算，ロード，ストア，分岐）
  - レジスタ番号 rs
  - レジスタ番号 rt
  - レジスタ番号 rd
  - 即値・オフセット
  - オペコード
  - ...
- 「**次PC計算**」部（分岐ユニットなどと呼ぶ）は，現在のPC値と2つのレジスタ値を入力として，次のPCの値を出力する組合せ回路である．内部では，分岐条件の判定と，分岐先アドレスの計算を行う

- 構成例: 教科書付録E章

練習問題

1. 2入力マルチプレクサ $m(a_0, a_1, s)$ を主加法標準形の論理式で表せ.

2. $m(a_0, a_1, s)$ のカルノー図をかき，できるだけ簡単な積和型の論理式で表せ．またその論理回路図を示せ.

3. 2ビット入力2進デコーダ2つとNOTゲート1つを使って，3ビット入力2進デコーダを作成せよ.

4. 全加算器の両出力 $s(a, b, c_{in})$, $c_{out}(a, b, c_{in})$ のカルノー図をかき，それぞれをできるだけ簡単な積和型の論理式で表せ.

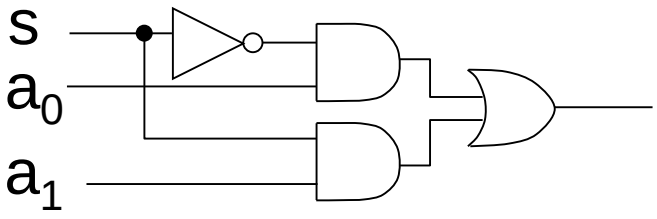
解答例

1. $m = \overline{a_0}\, a_1\, s + a_0\, \overline{a_1}\, \overline{s} + a_0\, a_1\, \overline{s} + a_0\, a_1\, s$

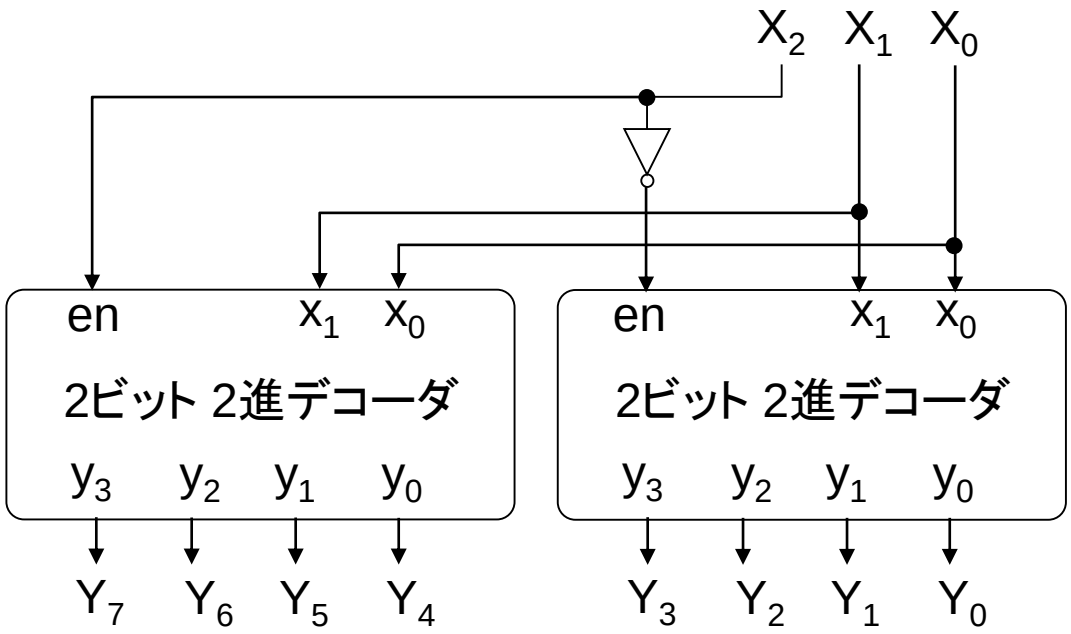
2.

m		a0 a1			
		00	01	11	10
s	0			1	1
	1		1	1	

$m = a_0\, \overline{s} + a_1\, s$



3. 出力の上位4本と下位4本をそれぞれの2ビットデコーダに担当させ，イネーブル信号で切り替えればよい



4.

s		a b			
		00	01	11	10
cin	0		1		1
	1	1		1	

cout		a b			
		00	01	11	10
cin	0			1	
	1		1	1	1

$$s = \overline{a}\,\overline{b}\,c_{\mathsf{in}} + \overline{a}b\,\overline{c_{\mathsf{in}}} + abc_{\mathsf{in}} + a\,\overline{b}\,\overline{c_{\mathsf{in}}}$$ (3入力 XOR)

$$c_{\mathsf{out}} = ab + bc_{\mathsf{in}} + c_{\mathsf{in}}a$$ (3入力多数決関数)