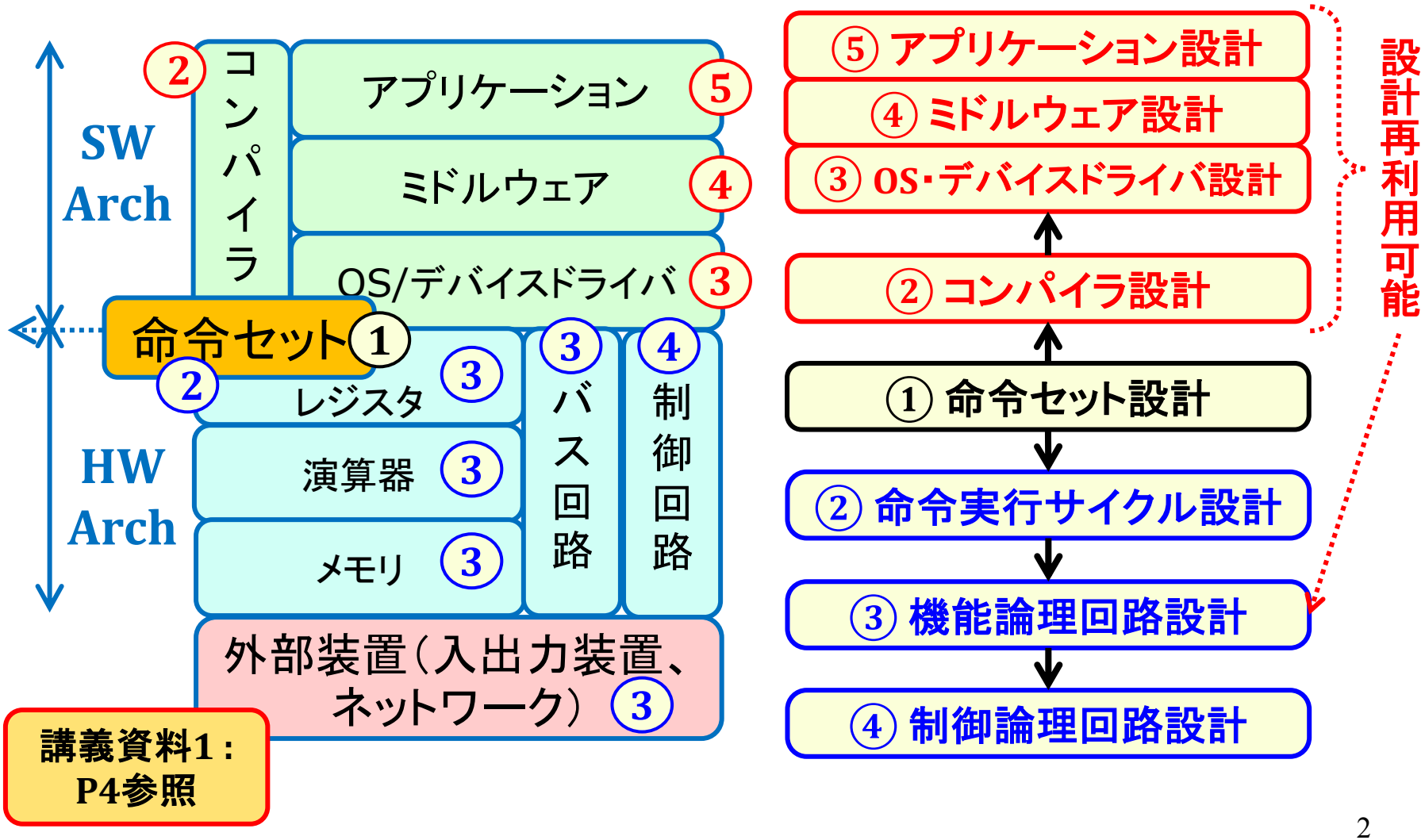
計算機論理設計
10. 計算機アーキテクチャ全体設計

一色 剛
工学院情報通信系
isshiki@ict.e.titech.ac.jp

計算機アーキテクチャ全体設計の流れ



講義資料1：P4参照

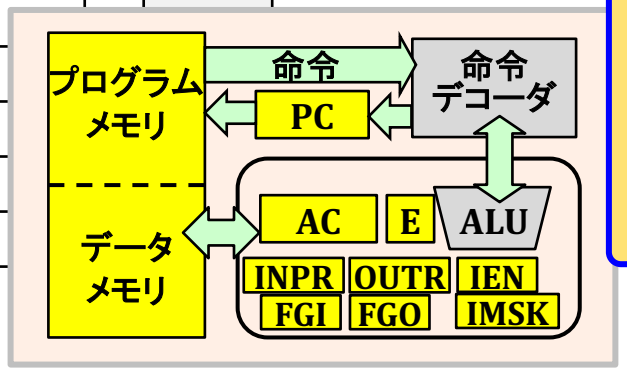
①命令セット設計：命令動作定義

命令	メモリ参照命令動作
AND	$AC \leftarrow AC \,\&\, M[ad]$
ADD	$AC \leftarrow AC + M[ad], E \leftarrow C_{out}(16)$
LDA	$AC \leftarrow M[ad]$
STA	$M[ad] \leftarrow AC$
BUN	$PC \leftarrow ad$
BSA	$M[ad] \leftarrow PC, PC \leftarrow ad + 1$
ISZ	$M[ad] \leftarrow M[ad] + 1,$ $IF(M[ad] + 1 = 0)\ THEN$ $PC \leftarrow PC + 1$

命令	入出力命令動作
INP	$AC(7{:}0) \leftarrow INPR, FGI \leftarrow 0$
OUT	$OUTR \leftarrow AC(7{:}0), FGO \leftarrow 0$
SKI	$IF\ (FGI = 1)\ THEN\ PC \leftarrow PC + 1$
SKO	$IF\ (FGO = 1)\ THEN\ PC \leftarrow PC + 1$
ION	$IEN \leftarrow 1$
IOF	$IEN \leftarrow 0$
IMK	$IMSK \leftarrow AC(1{:}0)$

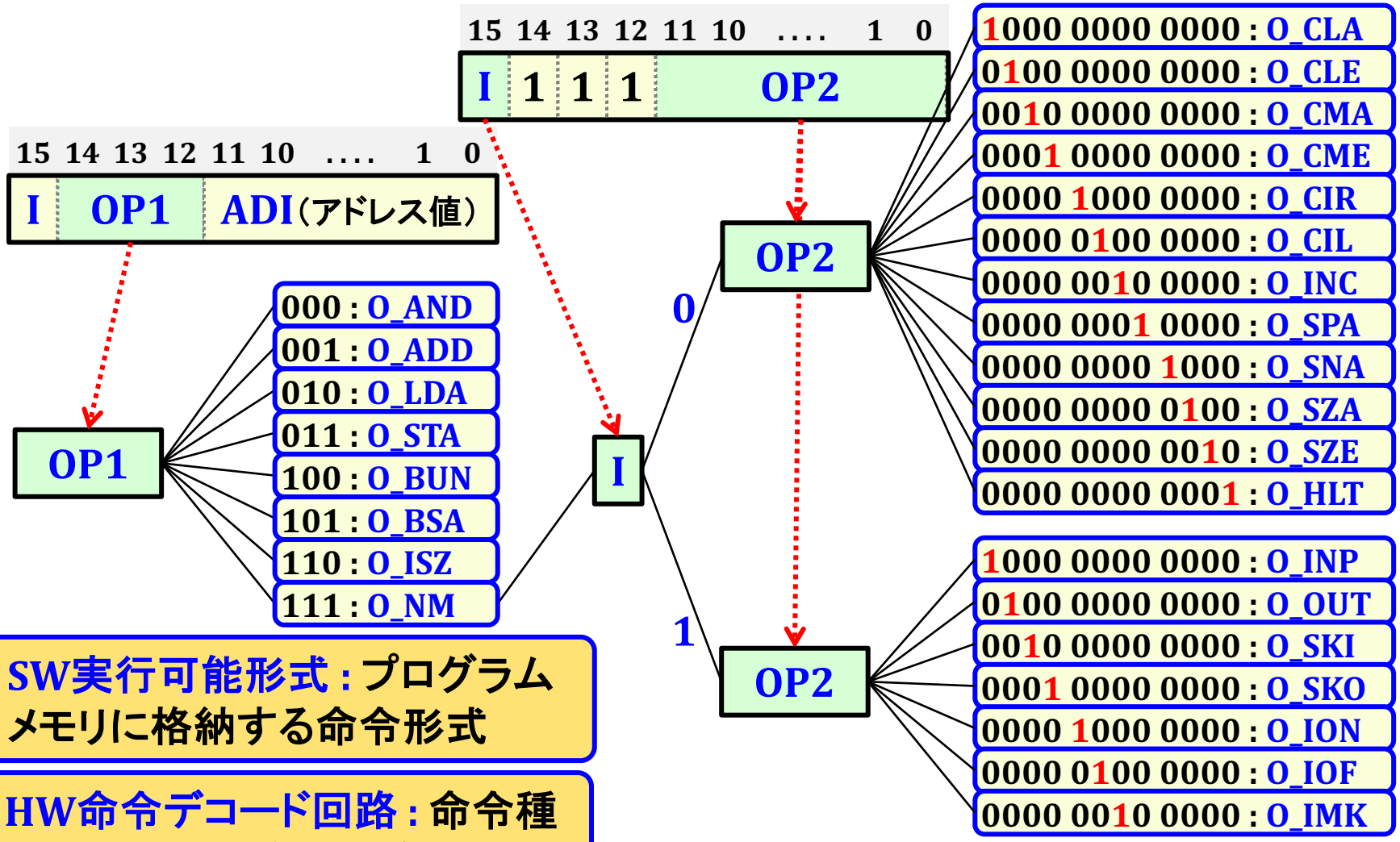
命令	レジスタ参照命令動作
CLA	$AC \leftarrow 0$
CLE	$E \leftarrow 0$
CMA	$AC \leftarrow \overline{AC}$
CME	$E \leftarrow \overline{E}$
CIR	$AC(14{:}0) \leftarrow AC(15{:}1), AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	$AC(15{:}1) \leftarrow AC(14{:}0), AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	$AC \leftarrow AC + 1$
SPA	$IF\ (AC(15) = 0)\ THEN\ PC \leftarrow PC + 1$
SNA	$IF\ (AC(15) = 1)\ THEN\ PC \leftarrow PC + 1$
SZA	$IF\ (AC = 0)\ THEN\ PC \leftarrow PC + 1$
SZE	$IF\ (E = 0)\ THEN\ PC \leftarrow PC + 1$
HLT	$HALT$

命令動作定義：レジスタ・メモリ間のデータ転送・演算動作を定義



簡易計算機構成：命令が直接作用するレジスタ・メモリのみ定義

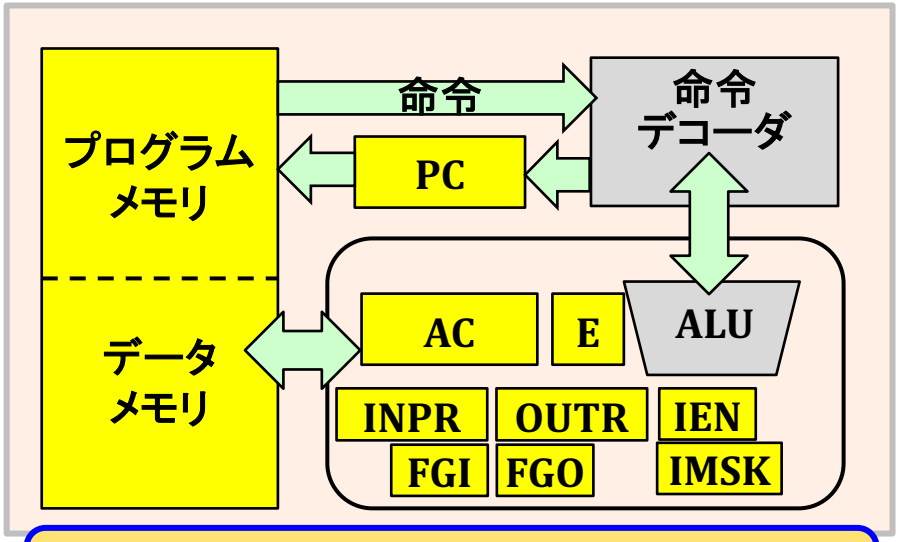
①命令セット設計：命令形式定義

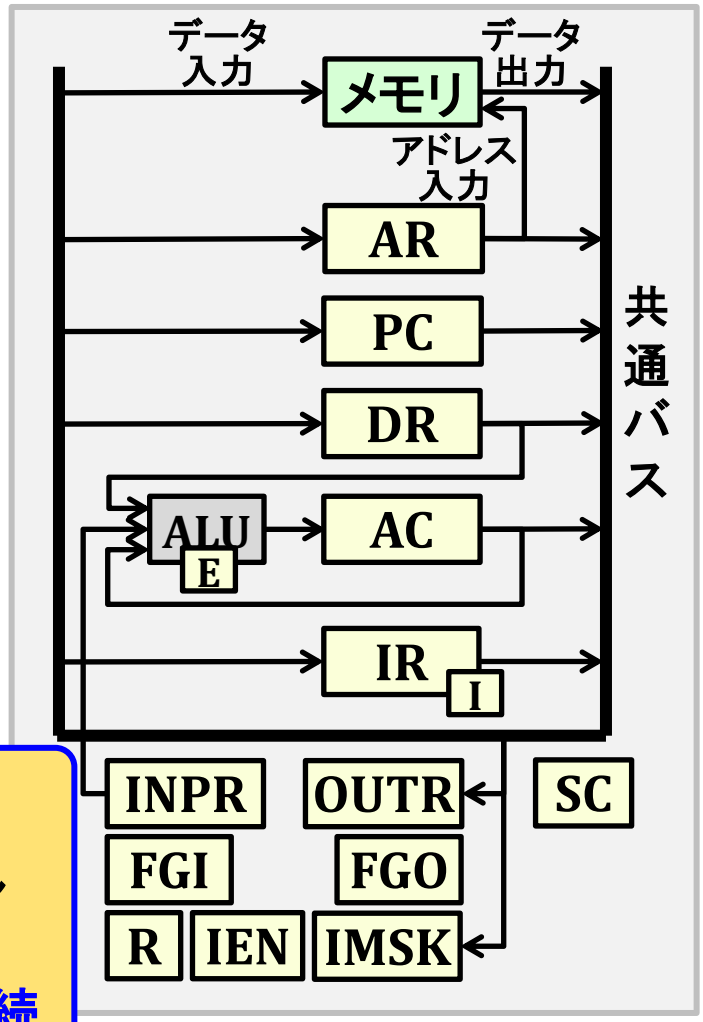


SW実行可能形式：プログラムメモリに格納する命令形式

HW命令デコード回路：命令種別を解釈する論理回路

②命令実行サイクル設計：計算機構成詳細化





簡易計算機構成：命令が直接作用するレジスタ・メモリのみ定義

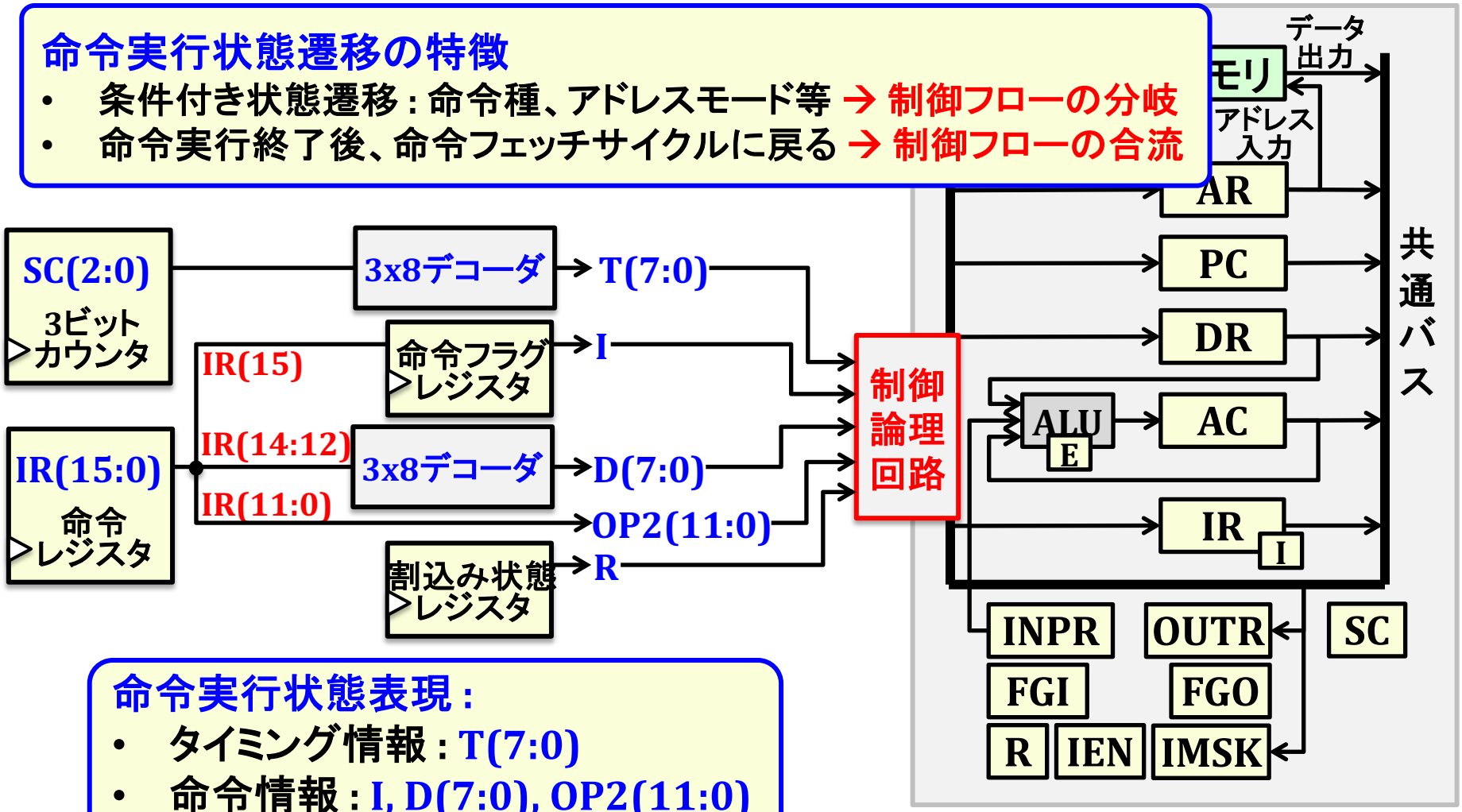
詳細計算機構成：

- 命令実行サイクルを実現するために必要なレジスタを追加：**AR, DR, IR**等
- データ転送経路定義：**共通バス、ALU/AC接続**

②命令実行サイクル設計：命令実行状態表現

命令実行状態遷移の特徴

- 条件付き状態遷移：命令種、アドレスモード等 → 制御フローの分岐
- 命令実行終了後、命令フェッチサイクルに戻る → 制御フローの合流



命令実行状態表現：

- タイミング情報：T(7:0)
- 命令情報：I, D(7:0), OP2(11:0)
- 割込み状態：R

②命令実行サイクル設計：命令動作詳細化

レジスタ転送記述形式

＜条件式＞：＜レジスタ転送式＞

割込み検知

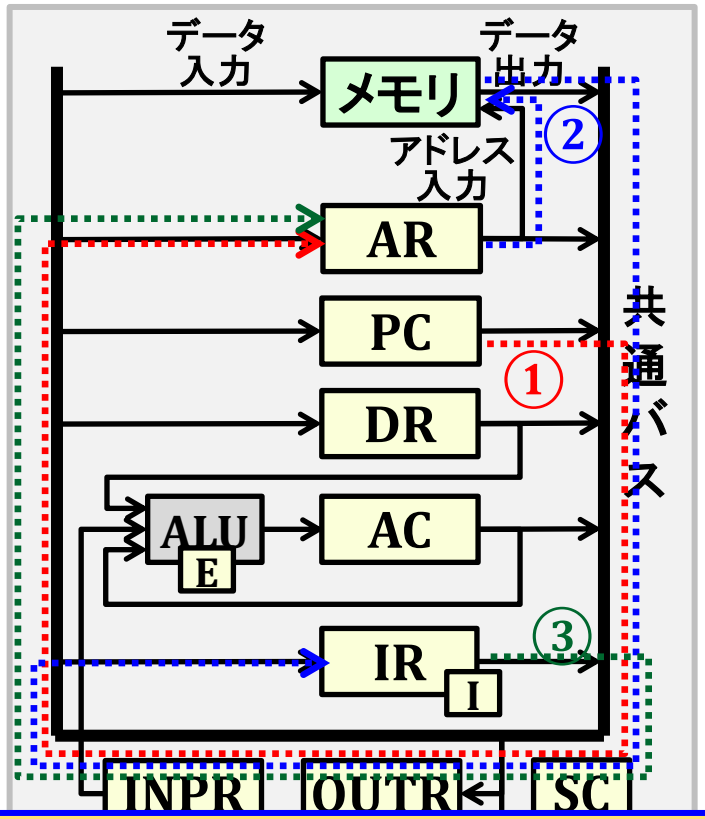
$INTR = IEN \cdot (IMSK(0) \cdot FGI + IMSK(1) \cdot FGO)$
$INTR \cdot \overline{T(0)} \cdot \overline{T(1)} \cdot \overline{T(2)}$ ： $R \leftarrow 1$

割込みサイクル

$R \cdot T(0)$:	$AR \leftarrow 0$
$R \cdot T(1)$:	$M[AR] \leftarrow PC, PC \leftarrow 0$
$R \cdot T(2)$:	$PC \leftarrow PC + 1, IEN \leftarrow 0,$ $R \leftarrow 0,\ \ SC \leftarrow 0$

命令フェッチサイクル

$\overline{R} \cdot T(0)$:	① $AR \leftarrow PC$
$\overline{R} \cdot T(1)$:	② $IR \leftarrow M[AR], PC \leftarrow PC + 1$
$\overline{R} \cdot T(2)$:	③ $I \leftarrow IR(15), AR \leftarrow IR(11:0)$



命令実行サイクルのレジスタ転送記述:

- 計算機の基本性能はこの時点で決まる → 各命令の実行サイクル数
- 計算機構成詳細化設計が大きく影響

②命令実行サイクル設計：命令動作詳細化

レジスタ参照命令実行サイクル

	$r = D(7) \cdot \bar{I} \cdot T(3)$
CLA	$r \cdot OP2(11):\ AC \leftarrow 0$
CLE	$r \cdot OP2(10):\ E \leftarrow 0$
CMA	$r \cdot OP2(9):\ AC \leftarrow \overline{AC}$
CME	$r \cdot OP2(8):\ E \leftarrow \overline{E}$
CIR	$r \cdot OP2(7):\ AC(14:0) \leftarrow AC(15:1),\ AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	$r \cdot OP2(6):\ AC(15:1) \leftarrow AC(14:0),\ AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	$r \cdot OP2(5):\ AC \leftarrow AC + 1$
SPA	$r \cdot OP2(4):\ IF\ (AC(15) = 0)\ THEN\ PC \leftarrow PC + 1$
SNA	$r \cdot OP2(3):\ IF\ (AC(15) = 1)\ THEN\ PC \leftarrow PC + 1$
SZA	$r \cdot OP2(2):\ IF\ (AC = 0)\ THEN\ PC \leftarrow PC + 1$
SZE	$r \cdot OP2(1):\ IF\ (E = 0)\ THEN\ PC \leftarrow PC + 1$
HLT	$r \cdot OP2(0):\ HALT$

入出力命令実行サイクル

	$p = D(7) \cdot I \cdot T(3)$
INP	$p \cdot OP2(11):\ AC(7:0) \leftarrow INPR, FGI \leftarrow 0$
OUT	$p \cdot OP2(10):\ OUTR \leftarrow AC(7:0), FGO \leftarrow 0$
SKI	$p \cdot OP2(9):\ IF\ (FGI = 1)\ THEN\ PC \leftarrow PC + 1$
SKO	$p \cdot OP2(8):\ IF\ (FGO = 1)\ THEN\ PC \leftarrow PC + 1$
ION	$p \cdot OP2(7):\ IEN \leftarrow 1$
IOF	$p \cdot OP2(6):\ IEN \leftarrow 0$
IMK	$p \cdot OP2(5):\ IMSK \leftarrow AC(1:0)$

非メモリ参照命令共通

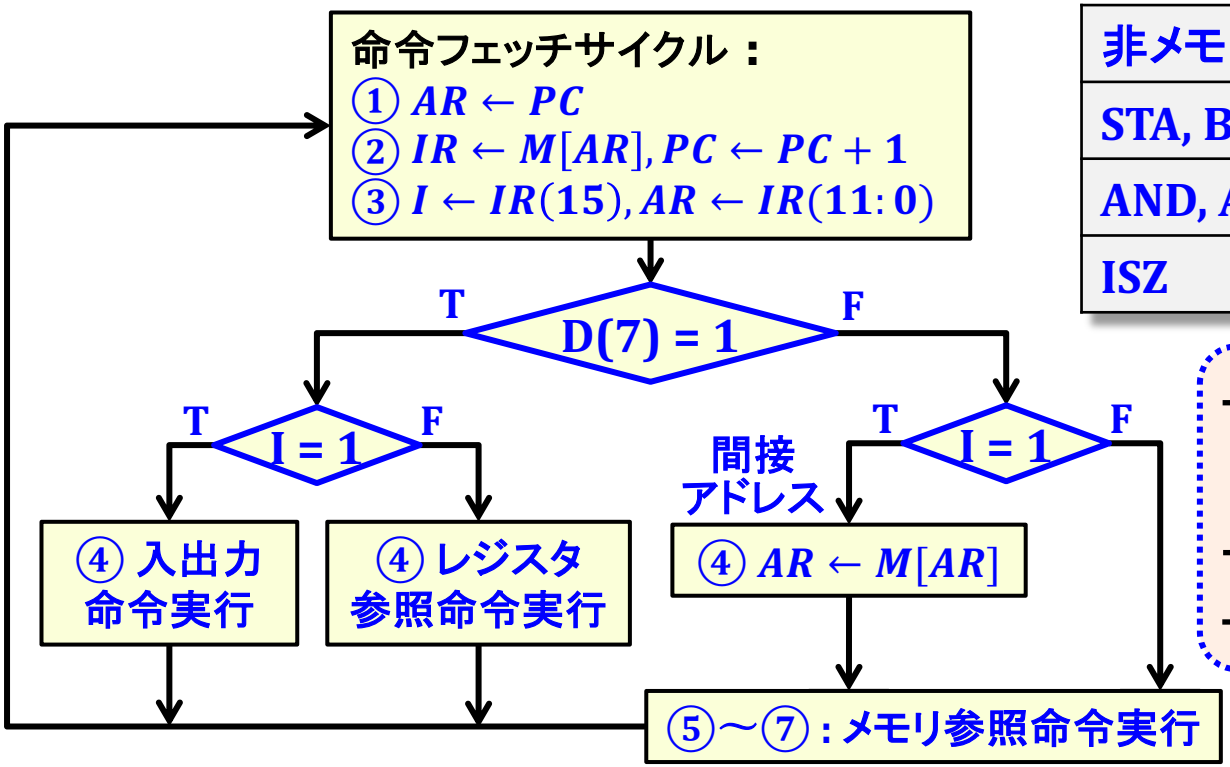
$D(7) \cdot T(3):\ SC \leftarrow 0$

②命令実行サイクル設計：命令動作詳細化

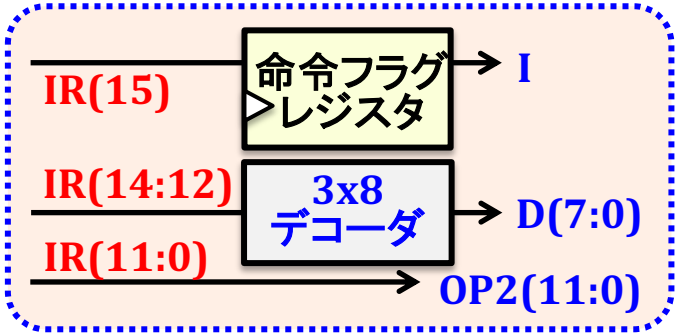
メモリ参照命令実行サイクル

	$\overline{D(7)} \cdot I \cdot T(3):$	$AR \leftarrow M[AR]$（間接アドレスサイクル）
AND	$D(0) \cdot T(4):$	$DR \leftarrow M[AR]$
	$D(0) \cdot T(5):$	$AC \leftarrow AC \,\&\, DR,\ SC \leftarrow 0$
ADD	$D(1) \cdot T(4):$	$DR \leftarrow M[AR]$
	$D(1) \cdot T(5):$	$AC \leftarrow AC + DR, E \leftarrow C_{out}(16),\ SC \leftarrow 0$
LDA	$D(2) \cdot T(4):$	$DR \leftarrow M[AR]$
	$D(2) \cdot T(5):$	$AC \leftarrow DR,\ SC \leftarrow 0$
STA	$D(3) \cdot T(4):$	$M[AR] \leftarrow AC,\ SC \leftarrow 0$
BUN	$D(4) \cdot T(4):$	$PC \leftarrow AR,\ SC \leftarrow 0$
BSA	$D(5) \cdot T(4):$	$M[AR] \leftarrow PC, AR \leftarrow AR + 1$
	$D(5) \cdot T(5):$	$PC \leftarrow AR,\ SC \leftarrow 0$
ISZ	$D(6) \cdot T(4):$	$DR \leftarrow M[AR]$
	$D(6) \cdot T(5):$	$DR \leftarrow DR + 1$
	$D(6) \cdot T(6):$	$M[AR] \leftarrow DR\,, IF(DR = 0)\ THEN\ PC \leftarrow PC + 1,\ SC \leftarrow 0$

計算機性能評価：命令実行サイクル数



非メモリ参照命令	4 cycles
STA, BUN	5 cycles
AND, ADD, LDA, BSA	6 cycles
ISZ	7 cycles



プログラム実行時間 = 実行命令数 × 平均命令実行サイクル数

- 命令実行サイクル設計が計算機の基本性能を決定する
- 命令実行サイクル設計改良による性能向上を後半で考える

命令セットの評価指標

講義資料4：
P16参照

- コード密度：高コード密度 → 小プログラムメモリ
  - ある機能を実現するための機械語命令列のサイズ
  - 命令の高機能化・複雑化や可変長命令によってコード密度は向上する

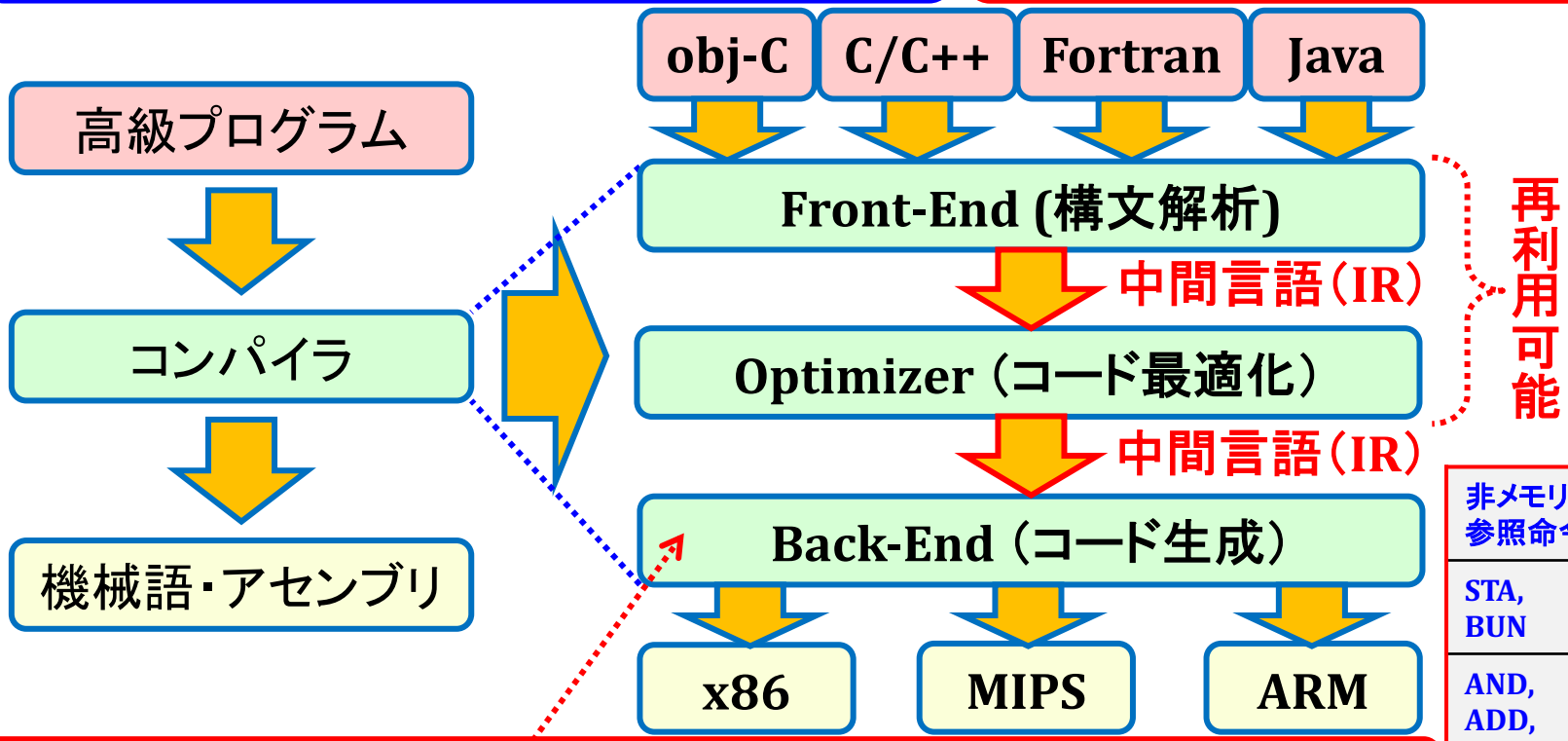
  → コード「品質」に大きく依存（コンパイラ、アセンブリプログラミング）
- 実行時間= 実行サイクル数×サイクル周期
  - 実行サイクル数 → 1命令の実行サイクル数に依存
  - サイクル周期 → 組合せ回路の最大論理遅延に依存

  → 命令実行制御を実現するハードウェアアーキテクチャに依存
- コンパイラ開発容易性：
  - コンパイラの「性能」→ 実行命令総数、コード密度
  - 複雑な機能を持つ命令セット用のコンパイラ開発が難しい
- 命令セット設計の難しさ：（命令セットの寿命が長い理由）
  - 計算機システム全体の完成まで、命令セットの「良し悪し」が評価しにくい
  - 命令セットの小さな修正が計算機システム設計全体に波及する

コンパイラ構成

コンパイラに必要な要件
- 多様な高級プログラミング言語に対応
- 多様な命令セットに対応

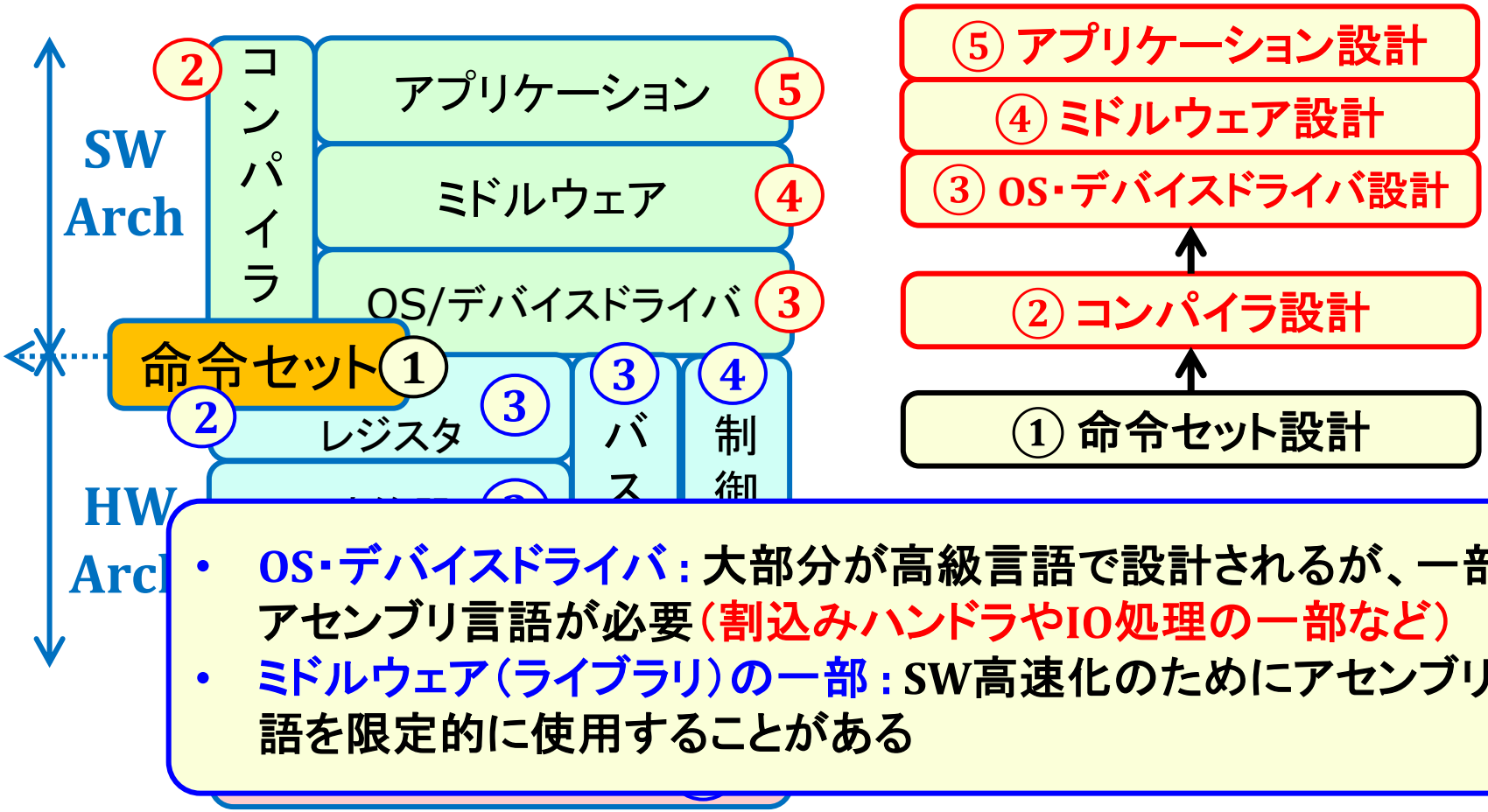
Front-End機能とOptimizer機能は命令セットに依存しない（再利用可能）



非メモリ参照命令	4 cycles
STA, BUN	5 cycles
AND, ADD, LDA, BSA	6 cycles
ISZ	7 cycles

- 命令セット設計・拡張 → Back-end（コード生成）の機能拡張が必要
- 各命令の実行サイクル数は、高品質なコード生成のために重要な情報

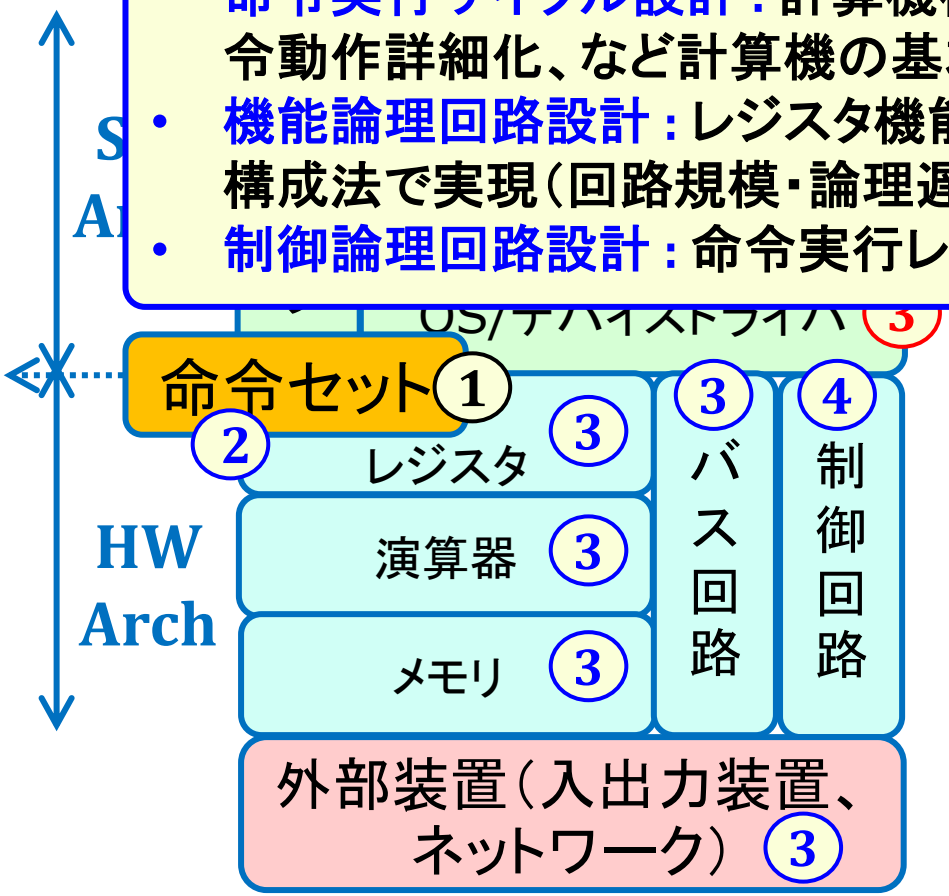
SWアーキテクチャ設計の流れ



- **OS･デバイスドライバ：**大部分が高級言語で設計されるが、一部アセンブリ言語が必要（割込みハンドラやIO処理の一部など）
- **ミドルウェア（ライブラリ）の一部：**SW高速化のためにアセンブリ言語を限定的に使用することがある

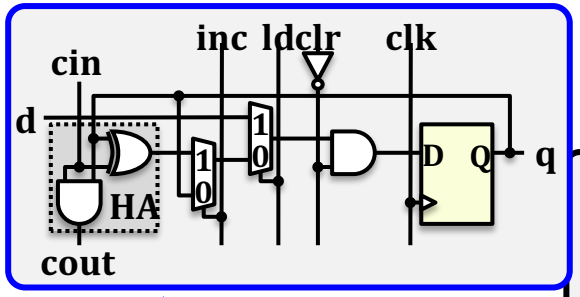
HWアーキテクチャ設計の流れ

- **命令実行サイクル設計：**計算機構成詳細化、命令実行状態表現、命令動作詳細化、など計算機の基本性能を決定する重要な設計工程
- **機能論理回路設計：**レジスタ機能・算術論理演算機能等を最適な回路構成法で実現（回路規模・論理遅延）
- **制御論理回路設計：**命令実行レジスタ転送記述から制御論理を導出

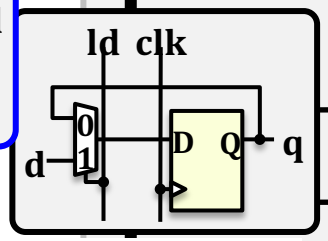


① 命令セット設計
↓
② 命令実行サイクル設計
↓
③ 機能論理回路設計
↓
④ 制御論理回路設計

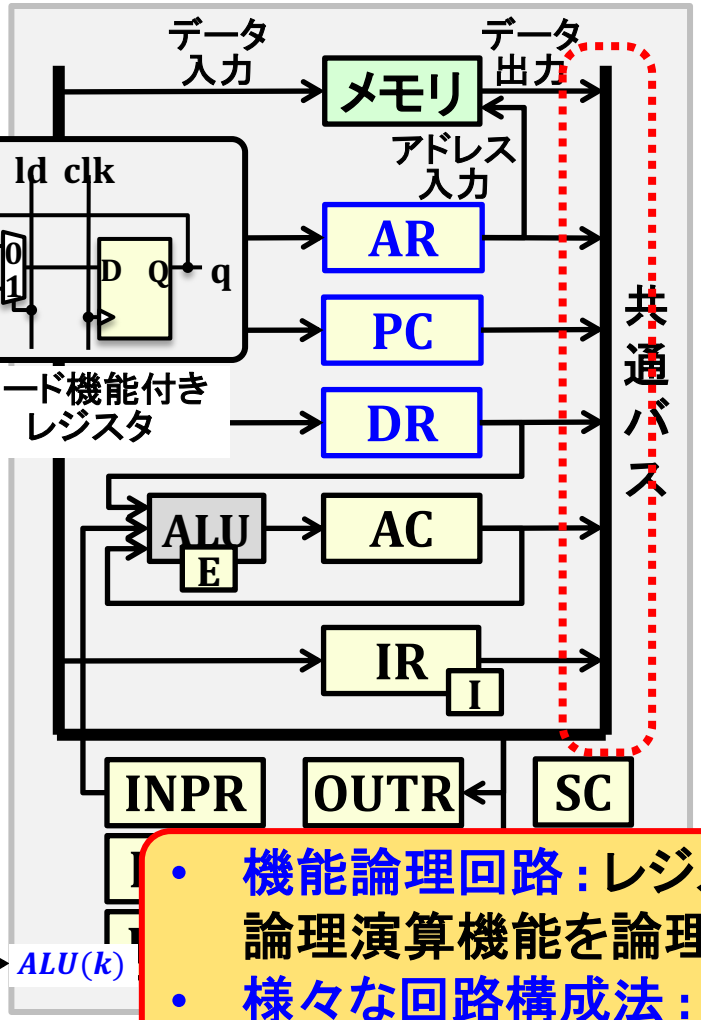
③機能論理回路設計

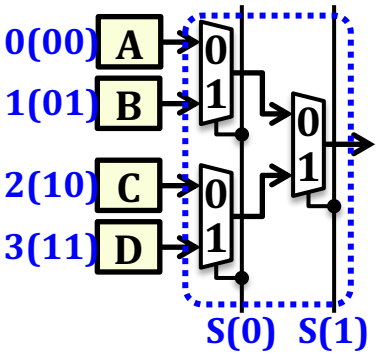


ロード・クリア・インクリメント機能付きレジスタ

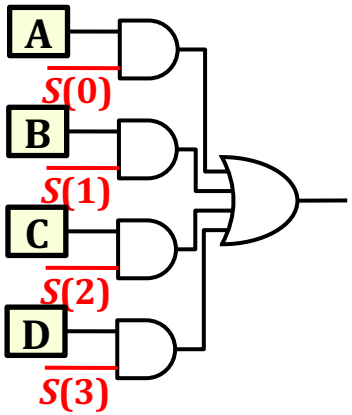


ロード機能付きレジスタ

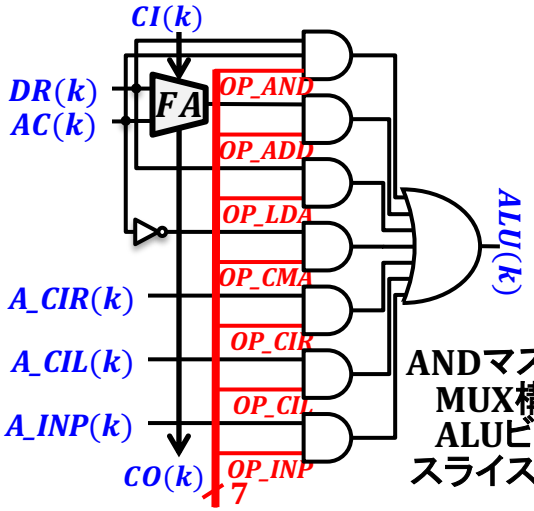




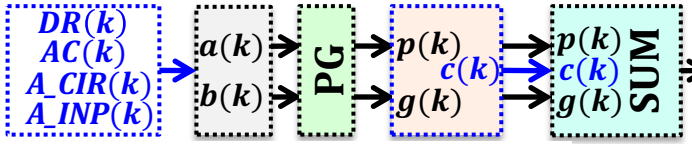
MUX-Tree型バス回路



ANDマスク型バス回路



ANDマスク型MUX構成ALUビットスライス回路



加算器機能共有型ALUビットスライス回路

- **機能論理回路**：レジスタ機能や算術論理演算機能を論理回路で実現
- **様々な回路構成法**：回路規模・論理遅延等で最適な構成法を選ぶ

④制御論理回路設計

メモリ書込み制御：

$$write_MEM = R \cdot T(1) + (D(3) + D(5)) \cdot T(4) + D(6) \cdot T(6)$$

割込みサイクル	$R \cdot T(1)$：	$M[AR] \leftarrow PC$
STA	$D(3) \cdot T(4)$：	$M[AR] \leftarrow AC$
BSA	$D(5) \cdot T(4)$：	$M[AR] \leftarrow PC$
ISZ	$D(6) \cdot T(6)$：	$M[AR] \leftarrow DR$

メモリ出力バス転送：

$$bus_MEM = \overline{R} \cdot T(1) + \overline{D(7)} \cdot I \cdot T(3) + (D(0) + D(1) + D(2) + D(6)) \cdot T(4)$$

命令フェッチ	$\overline{R} \cdot T(1)$：	$IR \leftarrow M[AR]$
間接アドレス	$\overline{D(7)} \cdot I \cdot T(3)$：	$AR \leftarrow M[AR]$
AND	$D(0) \cdot T(4)$：	$DR \leftarrow M[AR]$
ADD	$D(1) \cdot T(4)$：	$DR \leftarrow M[AR]$
LDA	$D(2) \cdot T(4)$：	$DR \leftarrow M[AR]$
ISZ	$D(6) \cdot T(4)$：	$DR \leftarrow M[AR]$

PCインクリメント制御：

$$PC_INC = R \cdot T(2) + \overline{R} \cdot T(1) + D(6) \cdot T(6) \cdot (DR = 0) + r \cdot (OP2(4) \cdot \overline{AC(15)} + OP2(3) \cdot AC(15) + OP2(2) \cdot (AC = 0) + OP2(1) \cdot \overline{E}) + p \cdot (OP2(9) \cdot FGI + OP2(8) \cdot FGO)$$

割込み	$R \cdot T(2)$：	$PC \leftarrow PC + 1$
命令フェッチ	$\overline{R} \cdot T(1)$：	$PC \leftarrow PC + 1$
ISZ	$D(6) \cdot T(6)$：	$IF\ (DR = 0)\ THEN\ PC \leftarrow PC + 1$
SPA	$r \cdot OP2(4)$：	$IF\ (AC(15) = 0)\ THEN\ PC \leftarrow PC + 1$
SNA	$r \cdot OP2(3)$：	$IF\ (AC(15) = 1)\ THEN\ PC \leftarrow PC + 1$
SZA	$r \cdot OP2(2)$：	$IF\ (AC = 0)\ THEN\ PC \leftarrow PC + 1$
SZE	$r \cdot OP2(1)$：	$IF\ (E = 0)\ THEN\ PC \leftarrow PC + 1$
SKI	$p \cdot OP2(9)$：	$IF\ (FGI = 1)\ THEN\ PC \leftarrow PC + 1$
SKO	$p \cdot OP2(8)$：	$IF\ (FGO = 1)\ THEN\ PC \leftarrow PC + 1$

命令動作レジスタ記述から、制御対象動作（書込み、出力転送、演算等）を抽出し、各実行条件の論理和として制御論理関数を導出する

条件付きレジスタ転送式（IF(cond) THEN形式）の場合、レジスタ転送条件式とcondの論理積を取る

入出力レジスタ・入出力命令

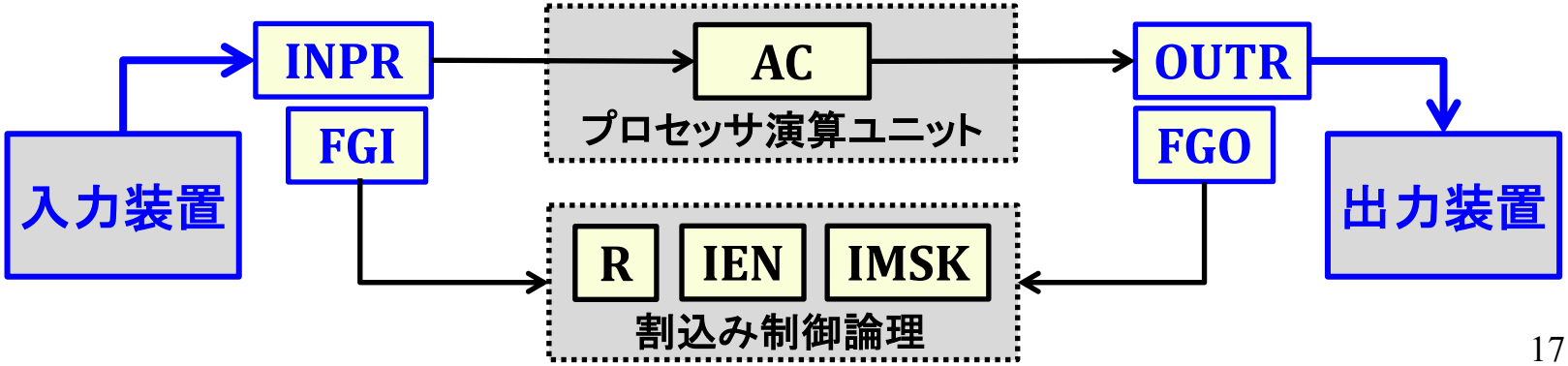
講義資料6：P3参照

INPR：入力レジスタ（8ビット）
FGI：入力フラグレジスタ（1ビット）
- FGI = 0：入力データなし状態
- FGI = 1：入力データあり状態

OUTR：出力レジスタ（8ビット）
FGO：出力フラグレジスタ（1ビット）
- FGO = 0：出力不可状態(busy)
- FGO = 1：出力可能状態(ready)

命令	動作
INP	$AC(7:0) \leftarrow INPR, FGI \leftarrow 0$
OUT	$OUTR \leftarrow AC(7:0), FGO \leftarrow 0$
SKI	$IF\ (FGI = 1)\ THEN\ PC \leftarrow PC + 1$
SKO	$IF\ (FGO = 1)\ THEN\ PC \leftarrow PC + 1$
ION	$IEN \leftarrow 1$
IOF	$IEN \leftarrow 0$
IMK	$IMSK \leftarrow AC(1:0)$

外部入出力装置から
FGO/FGIを
どのように制御するか？



入出力レジスタ制御

IN_DATA : 入力データ信号（8ビット）
IN_VLD : 入力有効状態信号（1ビット）
- IN_VLDは入力装置の出力信号
- 前提 : IN_VLD = 1の期間は1サイクルのみ

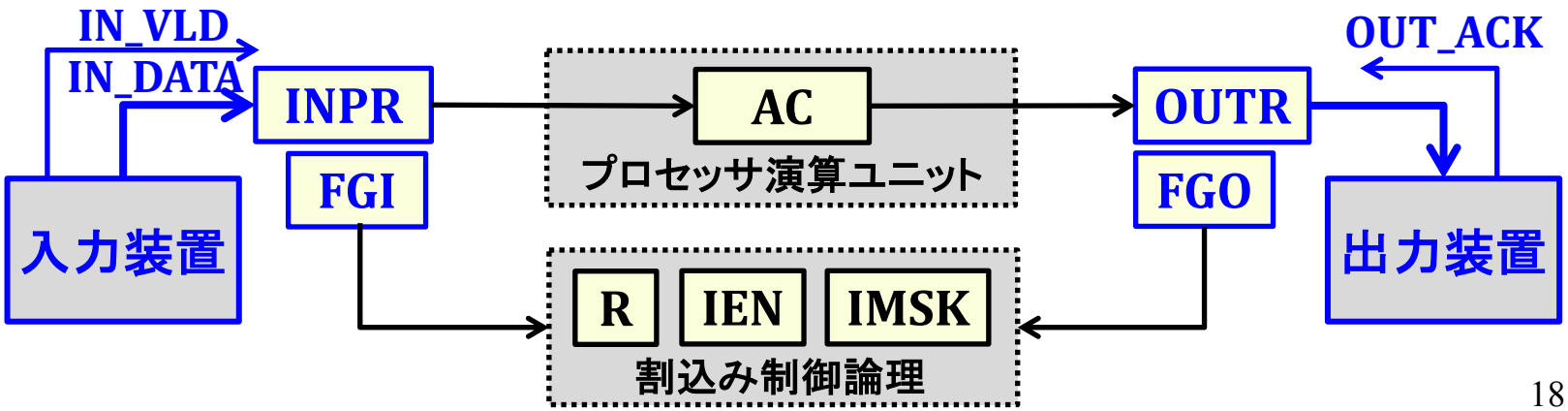
→ IN_DATAに有効なデータが転送されたことを示す

INP	$AC(7:0) \leftarrow INPR, FGI \leftarrow 0$
IN_VLD :	$INPR \leftarrow IN_DATA, FGI \leftarrow 1$

OUT_ACK : 出力転送状態信号（1ビット）
- OUT_ACKは出力装置の出力信号
- 前提 : OUT_ACK = 1の期間は1サイクルのみ

→ OUTRが出力装置に転送されたことを示す

OUT	$OUTR \leftarrow AC(7:0), FGO \leftarrow 0$
OUT_ACK :	$FGO \leftarrow 1$



入出力レジスタ制御

割込み検知

$INTR = IEN \cdot (IMSK(0) \cdot FGI + IMSK(1) \cdot FGO)$
$INTR \cdot \overline{T(0)} \cdot \overline{T(1)} \cdot \overline{T(2)}$: $R \leftarrow 1$

割込みサイクル

$R \cdot T(2)$: $IEN \leftarrow 0, R \leftarrow 0$

外部装置入力

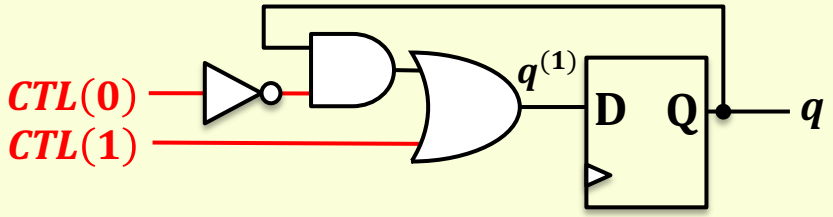
IN_VLD :	$INPR \leftarrow IN_DATA, FGI \leftarrow 1$
OUT_ACK :	$FGO \leftarrow 1$

入出力命令実行サイクル

	$p = D(7) \cdot I \cdot T(3)$	
INP	$p \cdot OP2(11)$:	$FGI \leftarrow 0$
OUT	$p \cdot OP2(10)$:	$FGO \leftarrow 0$
ION	$p \cdot OP2(7)$:	$IEN \leftarrow 1$
IOF	$p \cdot OP2(6)$:	$IEN \leftarrow 0$

1ビットレジスタ駆動回路:

- 制御入力（one-hot符号信号）: CTL(1:0)



CTL(1:0)	$q^{(1)}$
01	0
10	1
00	q

入出力レジスタ制御

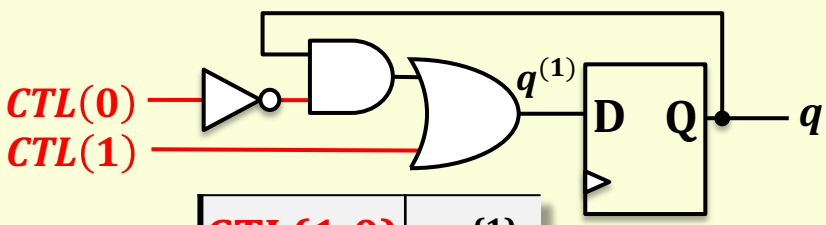
$INTR = IEN \cdot (IMSK(0) \cdot FGI + IMSK(1) \cdot FGO)$
$INTR \cdot \overline{T(0)} \cdot \overline{T(1)} \cdot \overline{T(2)}$: $R \leftarrow 1$

$R \cdot T(2)$: $IEN \leftarrow 0, R \leftarrow 0$

IN_VLD : $INPR \leftarrow IN_DATA, FGI \leftarrow 1$
OUT_ACK : $FGO \leftarrow 1$

	$p = D(7) \cdot I \cdot T(3)$	
INP	$p \cdot OP2(11)$:	$FGI \leftarrow 0$
OUT	$p \cdot OP2(10)$:	$FGO \leftarrow 0$
ION	$p \cdot OP2(7)$:	$IEN \leftarrow 1$
IOF	$p \cdot OP2(6)$:	$IEN \leftarrow 0$

$R_CTL(0) = R \cdot T(2)$
$R_CTL(1) = INTR \cdot \overline{T(0)} \cdot \overline{T(1)} \cdot \overline{T(2)}$
$IEN_CTL(0) = R \cdot T(2) + p \cdot OP2(6)$
$IEN_CTL(1) = p \cdot OP2(7)$
$FGI_CTL(0) = p \cdot OP2(11)$
$FGI_CTL(1) = IN_VLD$
$FGO_CTL(0) = p \cdot OP2(10)$
$FGO_CTL(1) = OUT_ACK$



CTL(1:0)	$q^{(1)}$
01	0
10	1
00	q

計算機全体の初期化と実行状態制御

計算機全体の初期化（RST = 1の時）:

- **PC初期化 :** 最初のプログラム実行番地は0x10
- その他初期化が必要なレジスタ: R,IEN,FGI,FGO,SC

リセット動作

$$RST: \quad PC \leftarrow 0x010,\ R \leftarrow 0,\ IEN \leftarrow 0,\ FGI \leftarrow 0,\ FGO \leftarrow 1,\ SC \leftarrow 0,\ S \leftarrow 1$$

レジスタ	初期値
PC	0x010
R	0
IEN	0
FGI	0
FGO	1
SC	0
S	1

HLT命令の実装 : 実行状態フラグレジスタ S を追加

- Sの初期値 : 1（実行状態）
- HLT命令動作 :

HLT	$r \cdot OP2(0):\quad S \leftarrow 0$

- タイミング生成器(SC)の制御 : S = 0の時にSC ← 0 となるように変更

HALT動作
（計算機停止状態）

$$\overline{S}: \quad SC \leftarrow 0$$

計算機全体の初期化と実行状態制御

$RST: \quad PC \leftarrow 0x010, R \leftarrow 0, IEN \leftarrow 0, FGI \leftarrow 0, FGO \leftarrow 1, SC \leftarrow 0, S \leftarrow 1$

HLT	$r \cdot OP2(0): \quad S \leftarrow 0$

$R_CTL(0) = R \cdot T(2) + RST$

$R_CTL(1) = INTR \cdot \overline{T(0)} \cdot \overline{T(1)} \cdot \overline{T(2)}$

$IEN_CTL(0) = R \cdot T(2) + p \cdot OP2(6) + RST$

$IEN_CTL(1) = p \cdot OP2(7)$

$FGI_CTL(0) = p \cdot OP2(11) + RST$

$FGI_CTL(1) = IN_VLD$

$FGO_CTL(0) = p \cdot OP2(10)$

$FGO_CTL(1) = OUT_ACK + RST$

$S_CTL(0) = r \cdot OP2(0)$

$S_CTL(1) = RST$

レジスタ	初期値
PC	0x010
R	0
IEN	0
FGI	0
FGO	1
SC	0
S	1

→ 次ページ

→ 次ページ

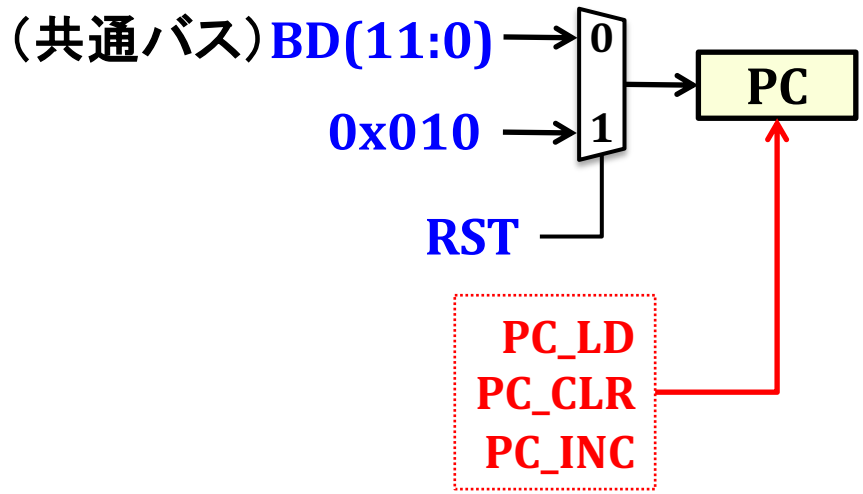
計算機全体の初期化と実行状態制御

$RST:\quad PC \leftarrow 0x010,\ R \leftarrow 0,\ IEN \leftarrow 0,\ FGI \leftarrow 0,\ FGO \leftarrow 1,\ SC \leftarrow 0,\ S \leftarrow 1$

種別	条件	動作
割込みサイクル	$R \cdot T(2):$	$SC \leftarrow 0$
AND	$D(0) \cdot T(5):$	$SC \leftarrow 0$
ADD	$D(1) \cdot T(5):$	$SC \leftarrow 0$
LDA	$D(2) \cdot T(5):$	$SC \leftarrow 0$
STA	$D(3) \cdot T(4):$	$SC \leftarrow 0$
BUN	$D(4) \cdot T(4):$	$SC \leftarrow 0$
BSA	$D(5) \cdot T(5):$	$SC \leftarrow 0$
ISZ	$D(6) \cdot T(6):$	$SC \leftarrow 0$
非メモリ参照命令	$D(7) \cdot T(3):$	$SC \leftarrow 0$
リセット	$RST:$	$SC \leftarrow 0$
HLT状態	$\overline{S}:$	$SC \leftarrow 0$

講義資料7：P21参照

$$SC_CLR = R \cdot T(2) + D(7) \cdot T(3) + \big(D(3) + D(4)\big) \cdot T(4) + (D(0) + D(1) + D(2) + D(5)) \cdot T(5) + D(6) \cdot T(6) + RST + \overline{S}$$



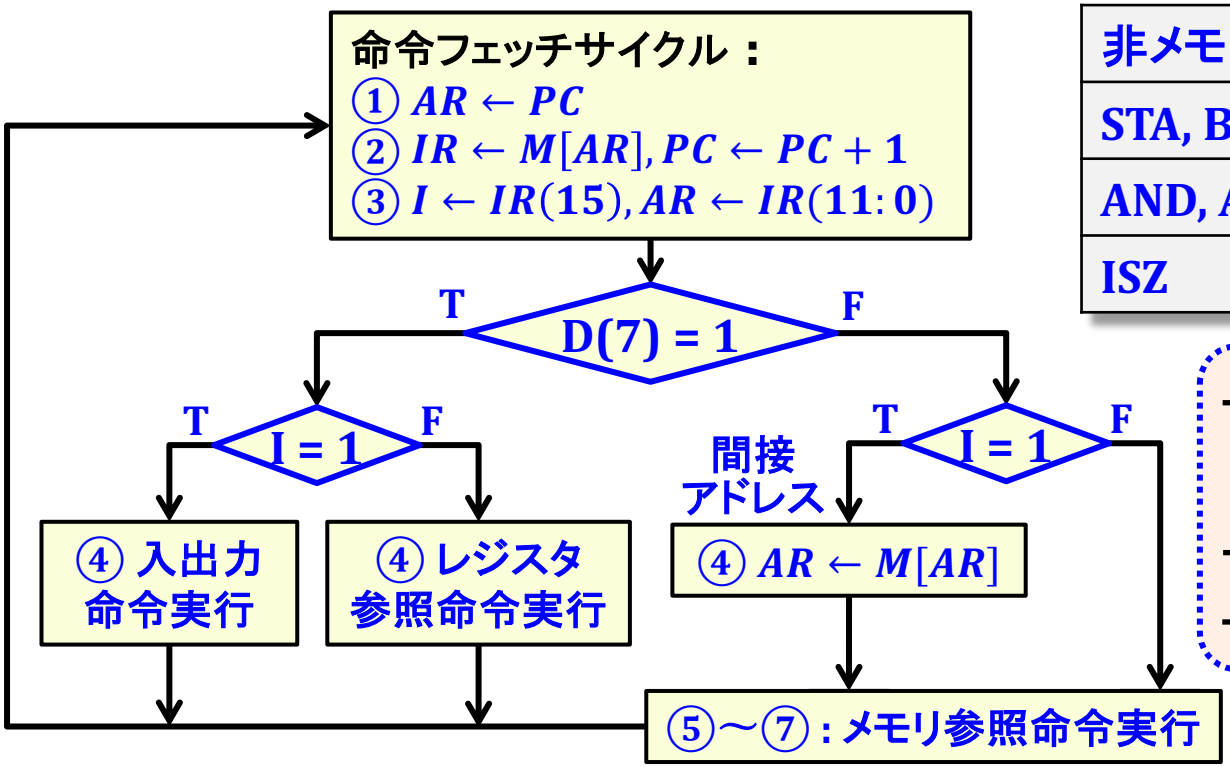
計算機ハードウェアアーキテクチャの最適化

命令セットの最適化（P11）: コード密度、プログラム実行時間、コンパイラ開発容易性、等の面で最適な命令セットを設計する
→ 一般的に非常に難しい、商用CPUの命令セットの寿命は非常に長い

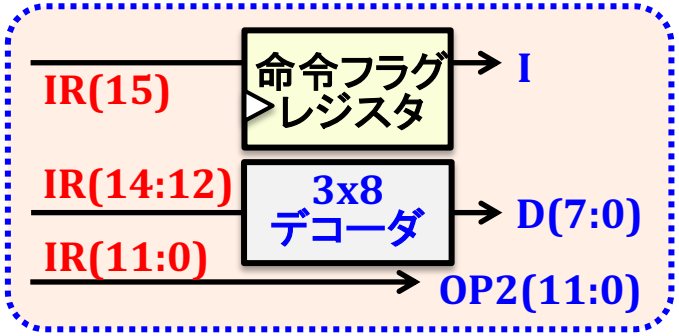
命令実行サイクル設計の最適化: 命令セット自体は変えずに、プログラム実行時間を最小なるような最適な命令実行サイクルを設計する
→ CPU技術は、この命令実行サイクル設計改良を中心に発展してきた
例：パイプライン式命令実行制御

回路設計の最適化： 同一の機能を実現する回路構成は複数存在し、その中から「最適」な回路構成を導出する
→ 回路規模・論理遅延に大きく影響する

計算機性能評価：命令実行サイクル数



非メモリ参照命令	4 cycles
STA, BUN	5 cycles
AND, ADD, LDA, BSA	6 cycles
ISZ	7 cycles

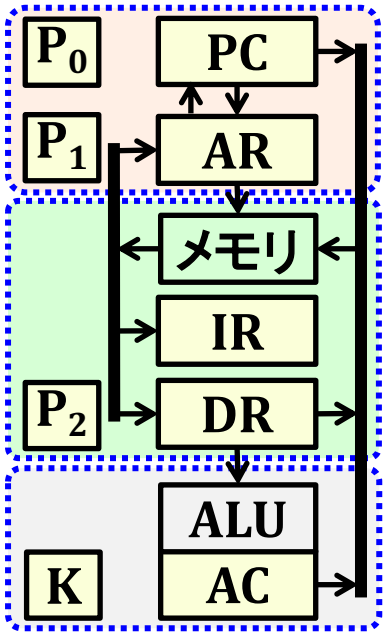


プログラム実行時間 = 実行命令数 × 平均命令実行サイクル数

- 命令実行サイクル設計が計算機の基本性能を決定する
- 命令実行サイクル設計改良による性能向上を後半で考える

命令実行サイクルのパイプライン制御

命令フェッチサイクルと命令実行サイクルをパイプライン化する



アドレスステージ：
$AR \leftarrow PC, PC \leftarrow PC + 1$
（AR, PC更新動作）

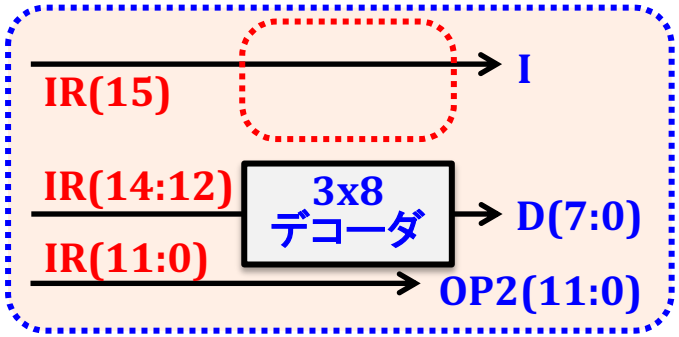
フェッチステージ：
$IR \leftarrow M[AR]$
（IR, DR更新動作）
（メモリアクセス動作）

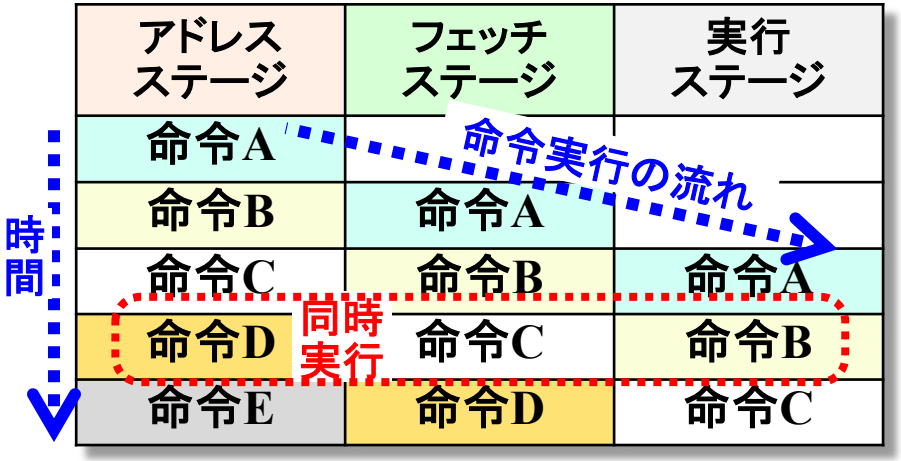
実行ステージ：
命令判別・実行

3ステージ（命令アドレス・命令フェッチ・命令実行）を同時に動作

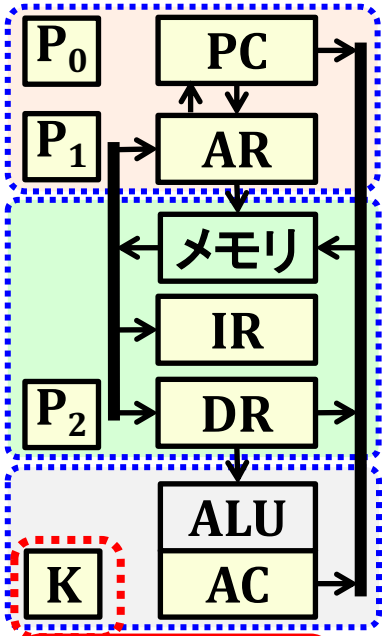
各レジスタ・メモリ書込みは、全て同じステージで行う

命令デコードの変更：命令フラグレジスタ (I) を取り除き、$I = IR(15)$とする
→ IRに命令を書き込んだ次のサイクルで命令判別が可能になる
→ 非メモリ参照命令は3サイクルに短縮





非メモリ参照命令のパイプライン動作



レジスタ転送動作	アドレスステージ	フェッチステージ	実行ステージ
$AR \leftarrow PC, PC \leftarrow PC + 1$	①		
$IR \leftarrow M[AR]$	次命令	②	
命令判別・実行	次々命令	次命令	③

非メモリ参照命令は3サイクル必要だが、3ステージを同時に動作させることで命令実行間隔は1サイクルに短縮される

アドレスステージ	フェッチステージ	実行ステージ
[A] $AR \leftarrow PC, PC \leftarrow PC + 1$		
[B] $AR \leftarrow PC, PC \leftarrow PC + 1$	[A] $IR \leftarrow M[AR]$	
[C] $AR \leftarrow PC, PC \leftarrow PC + 1$	[B] $IR \leftarrow M[AR]$	[A] 非スキップ命令
[D] $AR \leftarrow PC, PC \leftarrow PC + 1$	[C] $IR \leftarrow M[AR]$	[B] スキップ命令 ∵K設定
	[D] $IR \leftarrow M[AR]$	[C] スキップ命令 ∵K設定
		[D] 非スキップ命令

K :スキップフラグレジスタ

スキップフラグレジスタ K により次命令の実行条件を制御することでスキップ命令を実現

$IF(skipCond)\ THEN\ PC \leftarrow PC + 1$

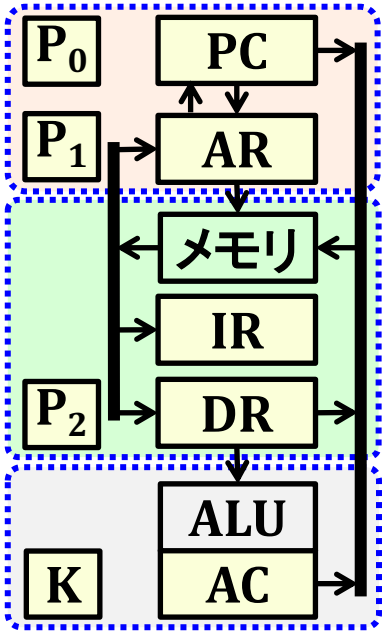
スキップ命令 : $K \leftarrow (skipCond)$

非メモリ参照命令実行 :

$IF(K)\ THEN\ K \leftarrow 0$ （スキップ状態解除）

$ELSE$ 命令実行 （非スキップ状態）

メモリ参照命令のパイプライン動作

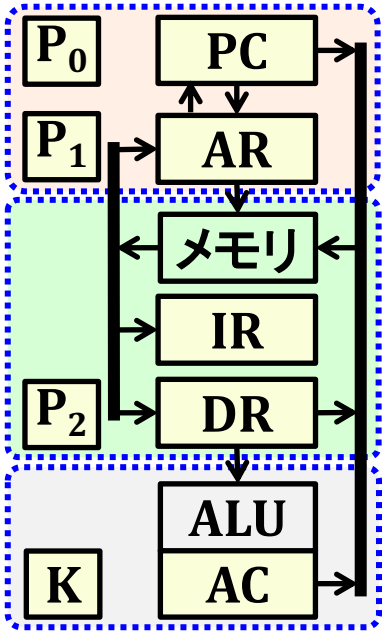


レジスタ転送動作	アドレスステージ	フェッチステージ	実行ステージ
$AR \leftarrow PC, PC \leftarrow PC + 1$	①		
$IR \leftarrow M[AR]$	次命令	②	
$AR \leftarrow IR(11:0), PC \leftarrow AR$	③	次命令	命令判別
$AR \leftarrow M[AR]$（I=1のみ）	④	④	
命令実行サイクル（⑤～⑦）			

メモリ参照命令：
メモリアクセス動作が次命令フェッチ動作と競合するため、**次命令フェッチ動作をキャンセル**する必要がある

	命令実行サイクル	アドレス	フェッチ	実行
AND, ADD, LDA	$DR \leftarrow M[AR]$	次命令	⑤	
	$AC \leftarrow op(AC, DR)$	次々命令	次命令	⑥
STA	$M[AR] \leftarrow AC$	次命令	⑤	
BUN	$PC \leftarrow AR$	⑤		
BSA	$M[AR] \leftarrow PC, AR \leftarrow AR + 1$	⑤	⑤	
	$PC \leftarrow AR$	⑥		
ISZ	$DR \leftarrow M[AR]$		⑤	
	$DR \leftarrow DR + 1$		⑥	
	$M[AR] \leftarrow DR, K \leftarrow (DR = 0)$	次命令	⑦	⑦

メモリ参照命令のパイプライン動作



$I = IR(15), OP2(11:0) = IR(11:0)$
$D(7:0) = DEC_{3\times8}(IR(14:12))$

実行ステージでメモリ参照命令が判別された場合$(D(7) = 0)$：
- 同時刻のアドレスステージ：$AR \leftarrow IR(11:0), PC \leftarrow AR$（前サイクルの次命令アドレスステージの$PC \leftarrow PC + 1$をキャンセルするため）
- 同時刻のフェッチステージ：無操作（IR更新を行わない）

アドレスステージ	フェッチステージ	実行ステージ
[A] $AR \leftarrow PC, PC \leftarrow PC + 1$		
[B] $AR \leftarrow PC, PC \leftarrow PC + 1$	[A] $IR \leftarrow M[AR]$	
[A] $AR \leftarrow IR(11:0), PC \leftarrow AR$	[B]（無操作）	[A] $(D(7) = 0)$
[A] $AR \leftarrow M[AR]$		[A]（無操作）
[B] $AR \leftarrow PC, PC \leftarrow PC + 1$	[A] $DR \leftarrow M[AR]$	[A]（無操作）
[C] $AR \leftarrow PC, PC \leftarrow PC + 1$	[B] $IR \leftarrow M[AR]$	[A] $AC \leftarrow op(AC, DR)$

AND, ADD, LDA命令の場合：フェッチステージで $DR \leftarrow M[AR]$ を実行する同時刻に、次命令のアドレスステージが実行可能

次命令のアドレスステージが開始可能になる時刻は、メモリ参照命令のパイプライン実行パターンに依存する

メモリ参照命令パイプライン動作パターン

命令実行間隔

AND, ADD, LDA

アドレス	フェッチ	実行
①		
次命令	②	
③	次命令	命令判別
④	④	
次命令	⑤	
次々命令	次命令	⑥

BUN

アドレス	フェッチ	実行
①		
次命令	②	
③	次命令	命令判別
④	④	
⑤		
次命令		

ISZ

アドレス	フェッチ	実行
①		
次命令	②	
③	次命令	命令判別
④	④	
	⑤	
	⑥	
次命令	⑦	⑦

命令実行間隔

STA

アドレス	フェッチ	実行
①		
次命令	②	
③	次命令	命令判別
④	④	
次命令	⑤	

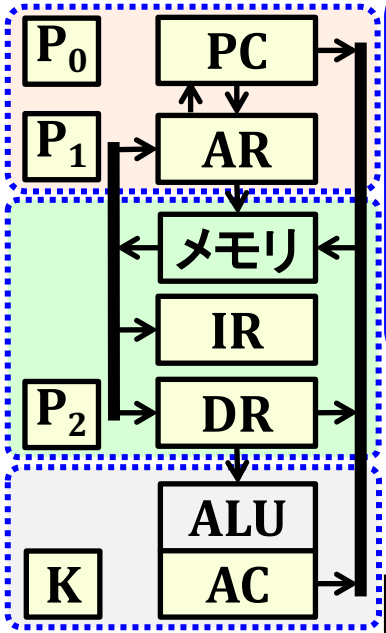
BSA

アドレス	フェッチ	実行
①		
次命令	②	
③	次命令	命令判別
④	④	
⑤	⑤	
⑥		
次命令		

BUN, BSA：最後の命令サイクル（$PC \leftarrow AR$）終了まで次命令アドレスステージが実行できない

その他のメモリ参照命令：命令実行が終了する前に次命令アドレスステージが実行可能になる

命令実行パイプライン制御方式



K :スキップフラグレジスタ

パイプライン制御は様々な方式が可能 → ここでは、これまでの「タイミング生成器（カウンタ）」をベースとした制御方式を考える

- P_0, P_1, P_2 : 3ビットカウンタ（ステージ毎に独立制御）
- 「実行ステージ」のメモリ参照命令判別までは $P_0 = P_1 = P_2 = 0$
- メモリ参照命令判別後、P_0, P_1, P_2 は自動インクリメント
- P_0, P_1, P_2 のリセット動作 : 前段から後段へ1クロック遅延で伝搬

$I = IR(15), OP2(11:0) = IR(11:0)$
$D(7:0) = DEC_{3\times8}(IR(14:12))$
$md = T_2(0) \cdot \overline{K} \cdot \overline{D(7)}$ （メモリ参照命令判別）

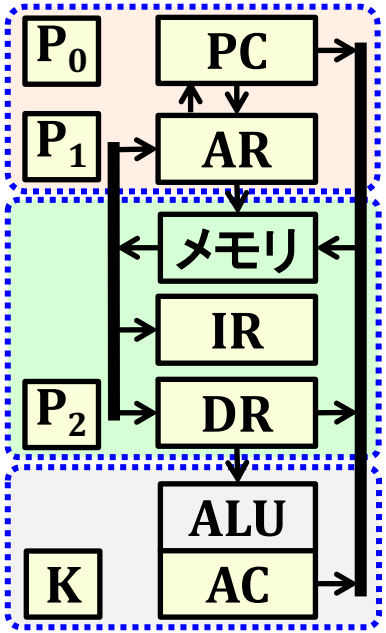
$T_0(7:0) = DEC_{3\times8}(P_0(2:0))$
$T_1(7:0) = DEC_{3\times8}(P_1(2:0))$
$T_2(7:0) = DEC_{3\times8}(P_2(2:0))$

レジスタ転送動作	アドレス	フェッチ	実行
$T_0(0) \cdot \overline{md}$: $AR \leftarrow PC, PC \leftarrow PC + 1, P_0 \leftarrow 0, P_1 \leftarrow 0$	①		
$T_1(0) \cdot \overline{md}$: $IR \leftarrow M[AR], P_2 \leftarrow 0$	次命令	②	
$T_2(0) \cdot D(7)$: 非メモリ参照命令実行	次々命令	次命令	③

スキップ命令 : $K \leftarrow (skipCond)$

非メモリ参照命令実行 :
$IF(K)\ THEN\ K \leftarrow 0$ （スキップ状態解除）
$ELSE$ 命令実行 （非スキップ状態）

メモリ参照命令パイプライン制御



レジスタ転送動作	アドレス	フェッチ	実行
$T_0(0)\cdot\overline{md}: AR \leftarrow PC, PC \leftarrow PC+1, P_0 \leftarrow 0, P_1 \leftarrow 0$	①		
$T_1(0)\cdot\overline{md}: IR \leftarrow M[AR], P_2 \leftarrow 0$	次命令	②	
$T_0(0)\cdot md: AR \leftarrow IR(11:0), PC \leftarrow AR(P_0{++}, P_1{++})$	③	次命令	命令判別
$T_0(1)\cdot I: AR \leftarrow M[AR]$	④	④	
$T_0(1)\cdot(D(0)+D(1)+D(2)+D(3)): P_0 \leftarrow 0$	次命令 アドレスステージ起動		

命令	レジスタ転送動作	アドレス	フェッチ	実行
AND, ADD, LDA	$T_1(2)\cdot(D(0)+D(1)+D(2)): DR \leftarrow M[AR]$	次命令	⑤	
	$T_2(3)\cdot(D(0)+D(1)+D(2)): AC \leftarrow op(AC, DR)$	次々命令	次命令	⑥
STA	$T_1(2)\cdot D(3): M[AR] \leftarrow AC$	次命令	⑤	
BUN	$T_0(2)\cdot D(4): PC \leftarrow AR, P_0 \leftarrow 0$	⑤		
BSA	$T_0(2)\cdot D(5): M[AR] \leftarrow PC, AR \leftarrow AR+1$	⑤	⑤	
	$T_0(3)\cdot D(5): PC \leftarrow AR, P_0 \leftarrow 0$	⑥		
ISZ	$T_1(2)\cdot D(6): DR \leftarrow M[AR]$		⑤	
	$T_1(3)\cdot D(6): DR \leftarrow DR+1, P_0 \leftarrow 0$		⑥	
	$T_1(4)\cdot D(6): M[AR] \leftarrow DR, K \leftarrow (DR=0)$	次命令	⑦	⑦

メモリ参照命令実行制御：各ステージ独立のカウンタ値を使う

メモリ参照命令パイプライン制御

$md = T_2(0) \cdot \overline{K} \cdot \overline{D(7)}$
（メモリ参照命令判別）

命令実行間隔

AND, ADD, LDA		
アドレス	フェッチ	実行
①		
次命令	②	
③	次命令	命令判別
④	④	
次命令	⑤	
次々命令	次命令	⑥
	次々命令	次命令

AND, ADD, LDA			
P_0	P_1	P_2	md
0			0
0	0		0
0	0	0	1
1	1	1	0
0	2	2	0
0	0	3	0
	0	0	0

$T_1(0) \cdot \overline{md}$: $IR \leftarrow M[AR]$
（$md = 1$の場合は無操作）

P_0, P_1, P_2
インクリメント
同時開始

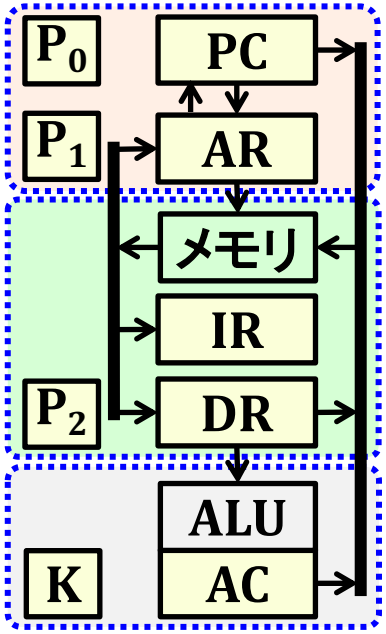
$T_0(1) \cdot (D(0) + D(1) + D(2) + D(3))$: $P_0 \leftarrow 0$

$T_0(0) \cdot \overline{md}$: $P_0 \leftarrow 0, P_1 \leftarrow 0$

$T_1(0) \cdot \overline{md}$: $P_2 \leftarrow 0$

P_0, P_1, P_2 のリセット動作：前段から後段へ1クロック遅延で伝搬

命令実行サイクルのパイプライン制御



アドレスステージ：
$AR \leftarrow PC, PC \leftarrow PC + 1$
（AR, PC更新動作）

フェッチステージ：
$IR \leftarrow M[AR]$
（IR, DR更新動作）
（メモリアクセス動作）

実行ステージ：
命令判別・実行

- 非メモリ参照命令実行間隔：1サイクル
- メモリ参照命令：AND, ADD, LDA, STA, ISZ命令は、次命令とオーバーラップして実行できるため、実行間隔が短縮

逐次型命令実行制御

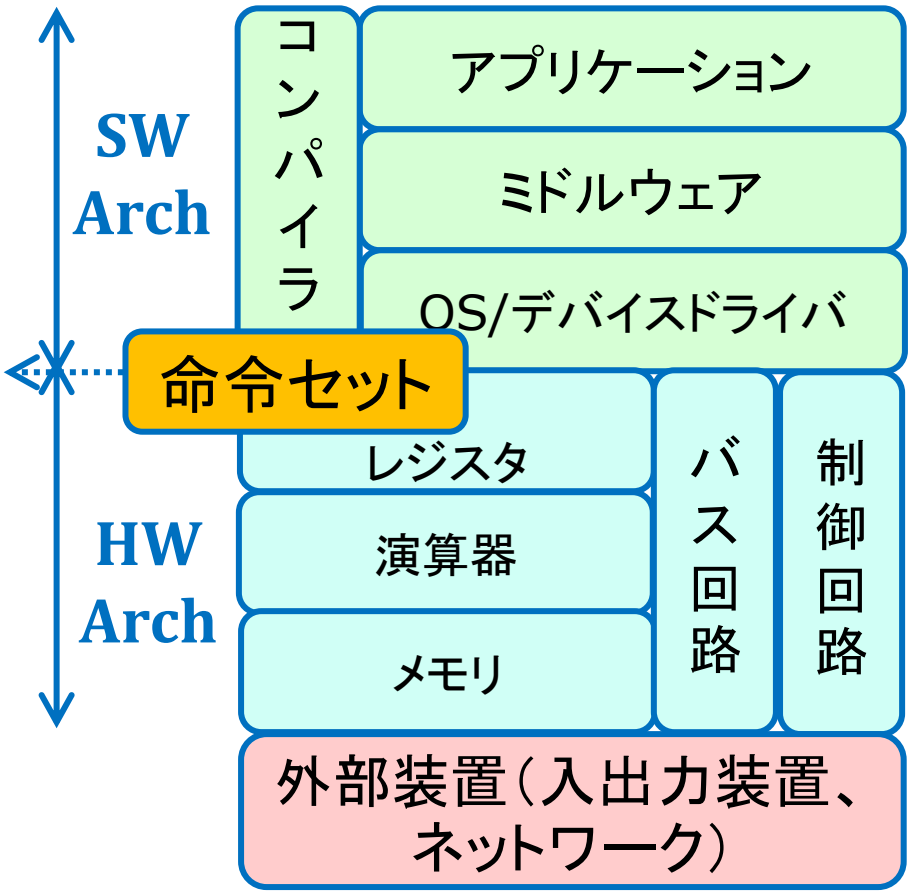
非メモリ参照命令	4 cycles
STA, BUN	5 cycles
AND, ADD, LDA, BSA	6 cycles
ISZ	7 cycles

パイプライン型命令実行制御

非メモリ参照命令	1 cycles
AND, ADD, LDA, STA	4 cycles
BUN	5 cycles
BSA, ISZ	6 cycles
スキップされた命令	1 cycles

現在の商用CPUは非常に複雑なパイプライン制御構造を持つ → 詳細は「計算機アーキテクチャ」講義で解説する

「計算機論理設計」第1部まとめ



命令セット：計算機の基本動作を定義した手続き（命令の集合）

- **SW：**実行可能形式（実行バイナリ）を定義
- **HW：**論理回路設計の「仕様」

命令実行サイクル 設計：計算機の基本性能を決定する重要な設計工程 → **CPU発展の主要部分**

論理設計：様々な回路構成法が存在する中から回路規模・論理遅延において最適な回路構成を導出する