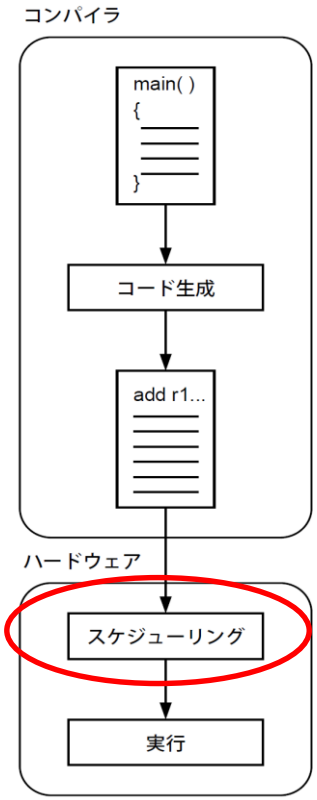
内容

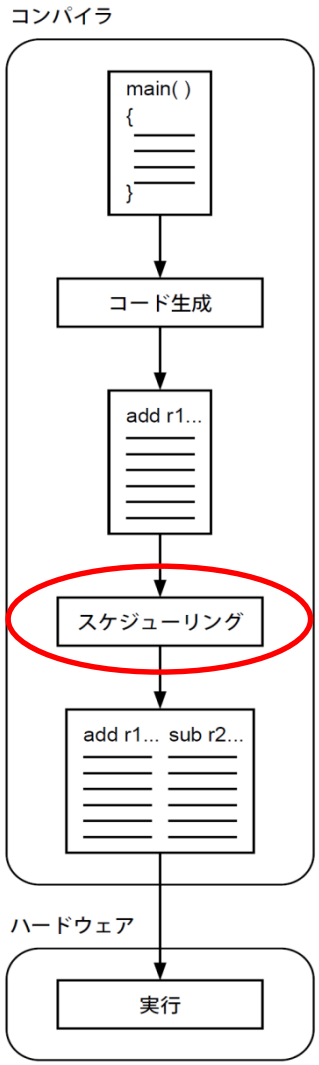
- VLIW (Very Long Instruction Word）プロセッサ
  - 静的命令スケジューリング
  - ハードウェア
  - スーパスカラ・プロセッサとの比較

- 命令スケジューリング
  - スケジューリング・アルゴリズムの分類
  - 局所スケジューリング：リスト・スケジューリング

静的命令スケジューリング



スーパスカラ



VLIW

VLIWコード

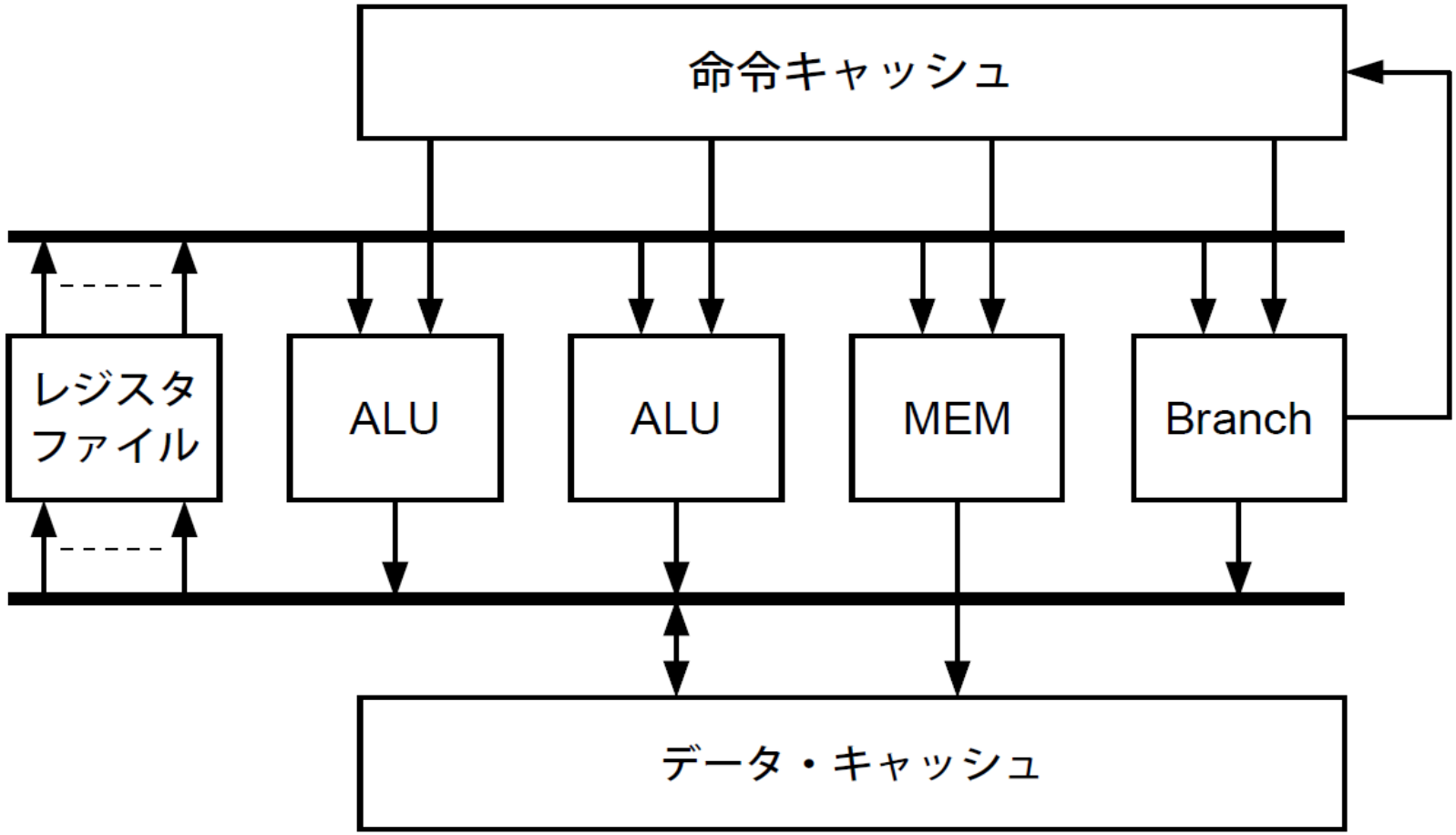
```
i1: r3 = r4 + 1
i2: r1 = load (r2)
i3: r1 = r1 < r3
i4: r5 = r2 + r6
i5: beq r4, L
```

ALU	ALU	branch	MEM
i1:r3=r4+1	i4:r5=r2+r6	nop	i2:r1=load(r2)
i3:r1=r1<r3	nop	i5:beq r4,L	nop

VLIWコード

- 機能ユニットに対応するスロットに命令を入れる
- 実行する命令がないときは、nop（no-operation）を入れる

VLIWハードウェア



スーパスカラ・プロセッサとの比較

観点	スーパスカラ	VLIW
ハードウェア量	大	小
ハードウェアの複雑さ	大	小
スケジューリング・アルゴリズム	簡素	高度
命令ウィンドウ	小	大
バイナリ互換	あり	なし

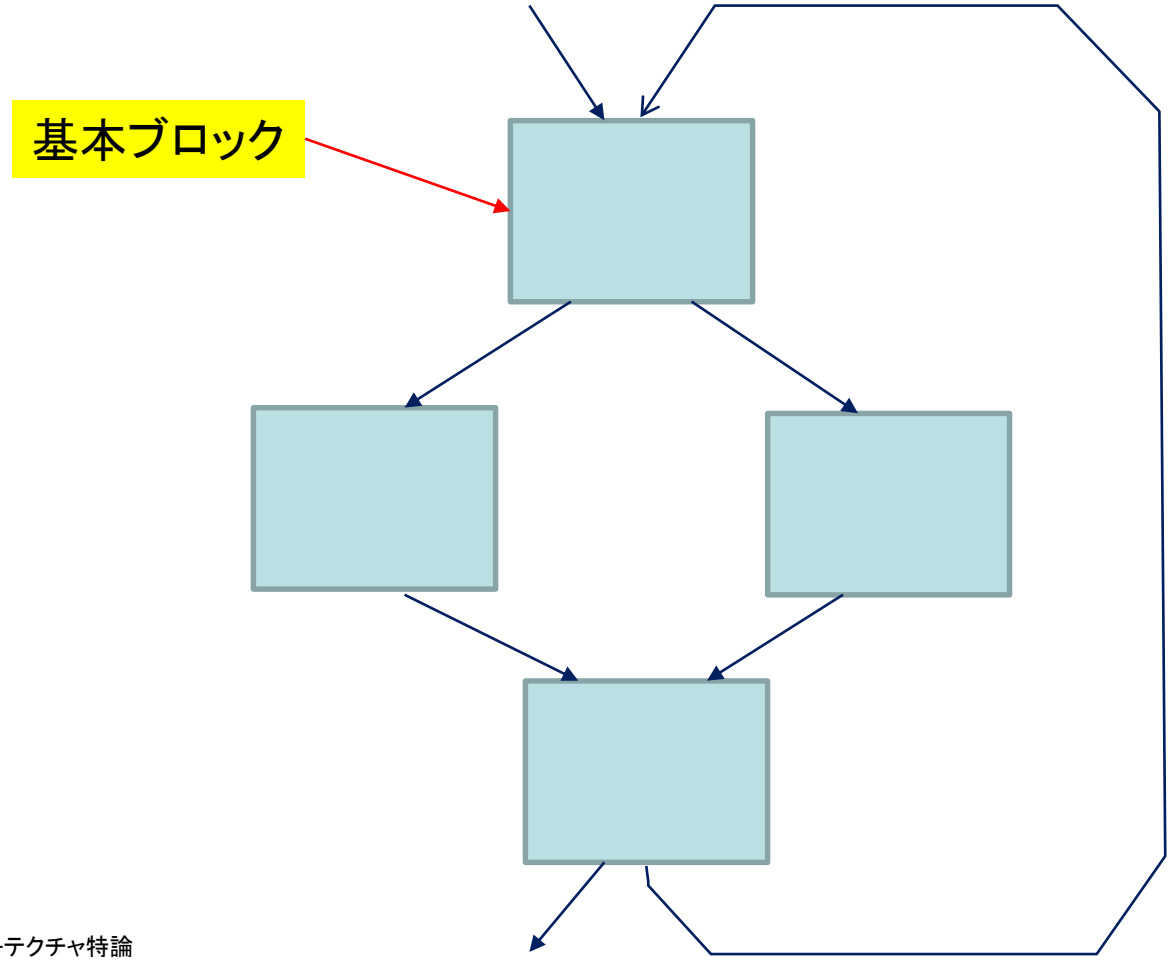
- スケジューリング・アルゴリズム
  - 限られた情報 vs. 豊富な情報 (e.g., クリティカル・パス情報)
  - 動的情報 vs. 静的情報 (e.g., キャッシュ・ミス)
- 命令ウィンドウ
  - 分岐予測制約 vs. コンパイラ複雑度制約

Pentium 4 vs. TM5400

プロセッサ	Intel Pentium 4	Transmeta TM5400
クロック速度	1.5GHz	700MHz
キャッシュ(I/D/L2)	12K/8K/256K	64K/64K/256K
製造プロセス	0.18μm	0.18μm
トランジスタ数	42M	2.8M
チップ面積	217mm^2	73mm^2

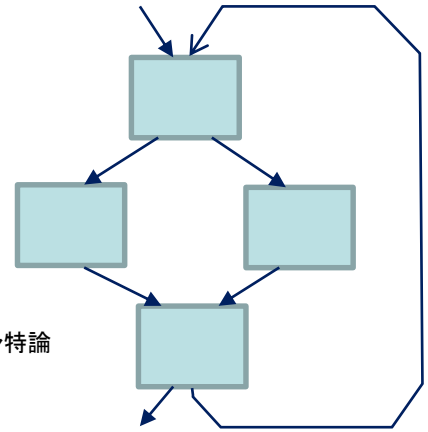
制御フロー・グラフ

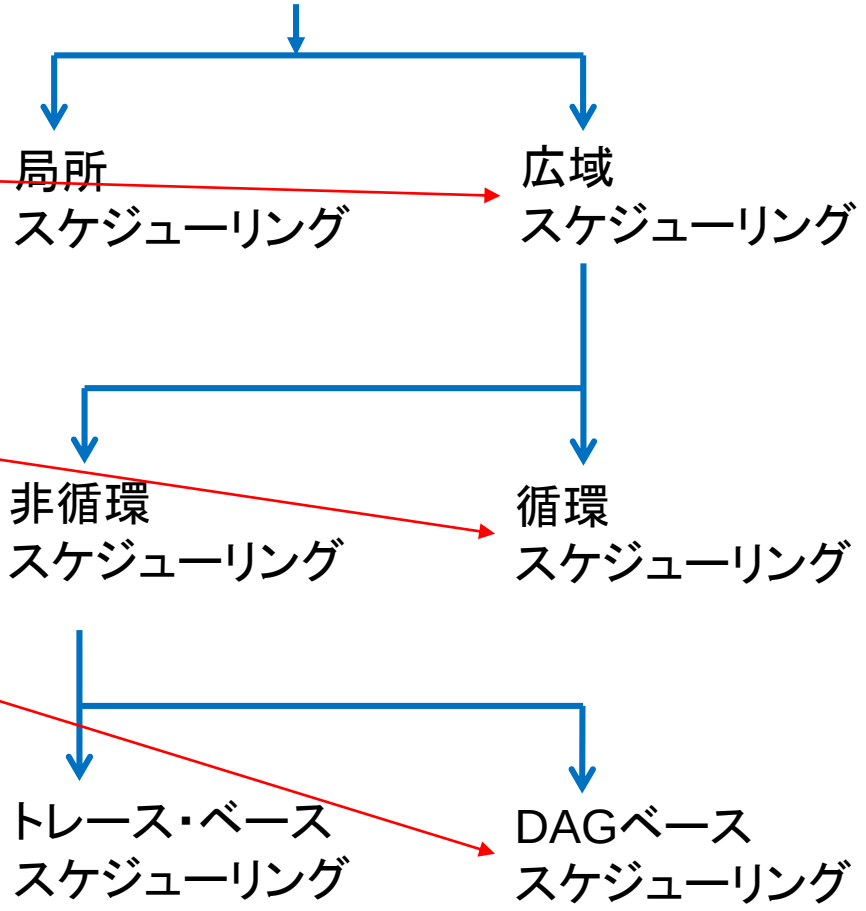
if-then-elseのループ



スケジューリング・アルゴリズムの分類

- 基本ブロック内
- 基本ブロックを超える
- ループがない
- ループ
- 単一制御フロー
- 複数制御フロー





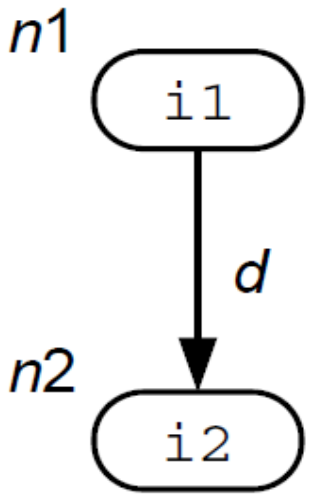
局所命令スケジューリング

- 基本ブロック内の命令のスケジューリング
  - データ依存
  - 資源制約
  - 制御フローは考えなくてよい

- 手順
  - 先行制約グラフ
    - 守るべき順序関係を表す
  - スケジュール

先行制約グラフ

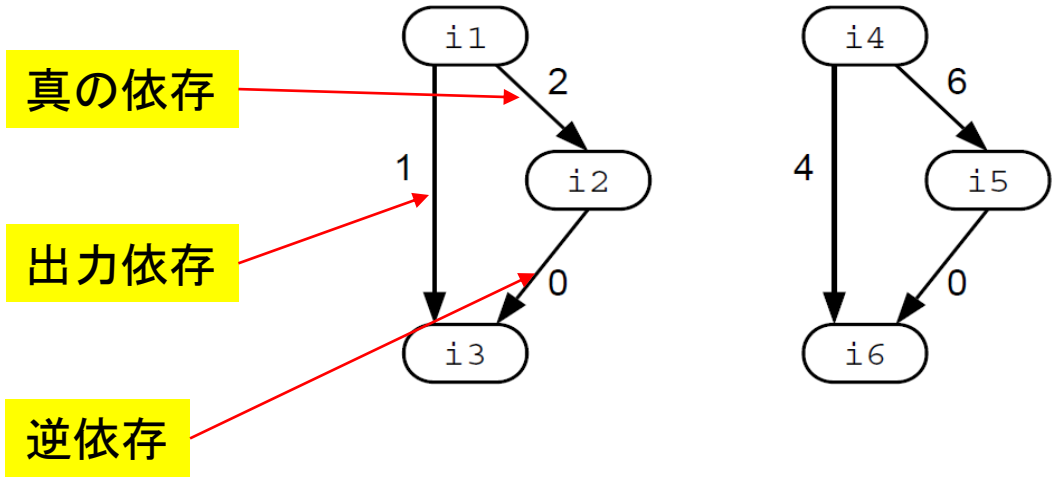
- G = (N,E)
  - N: 命令
  - E: 依存
  - Eのラベル: 遅延



先行制約グラフの例

```
i1: r1 = load A
i2: r3 = r1 + 1
i3: r1 = r4 + 2
i4: f5 = f2 x f4
i5: store B = f5
i6: f5 = f6 + f7
```

- 実行レイテンシ→load:2, fmul:6, fadd:3, other:1

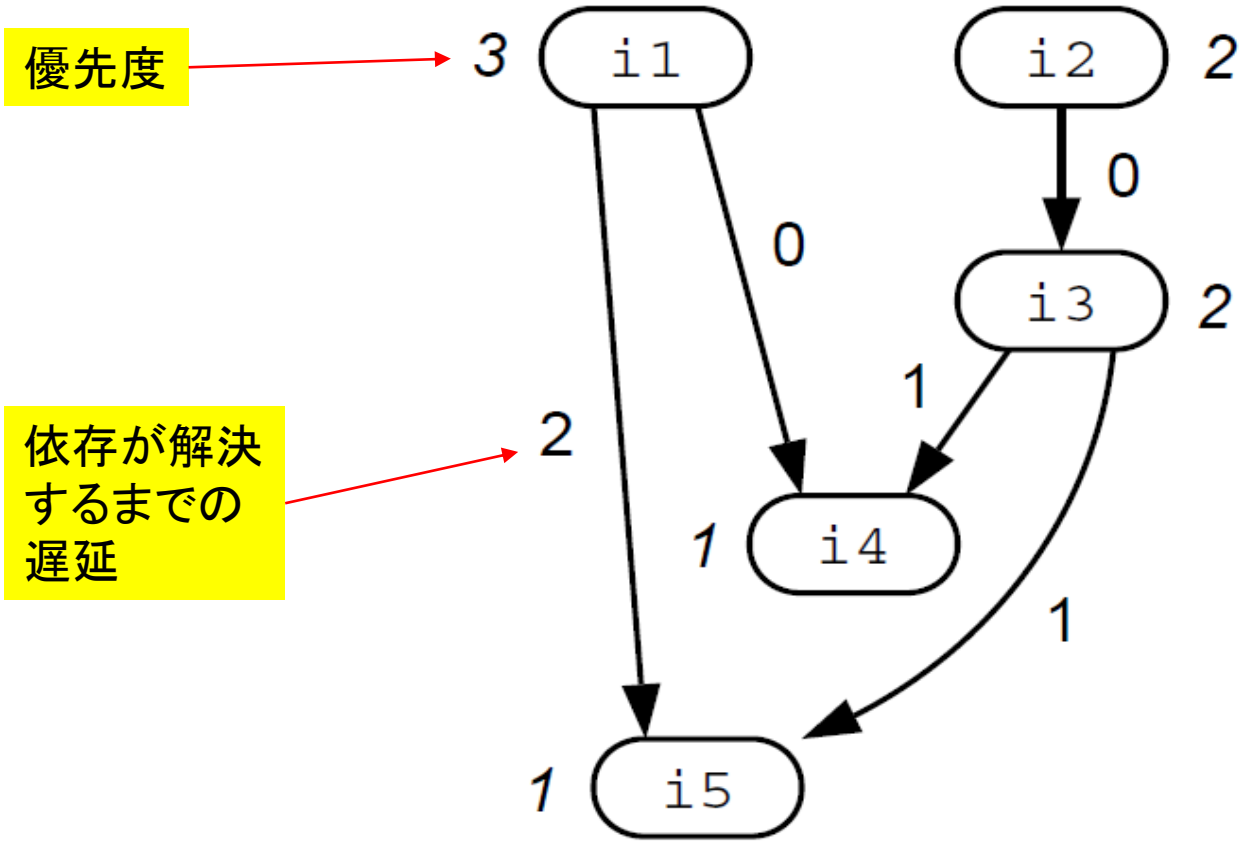


リスト・スケジューリング

- トポロジカル・ソート
  - ノード*i*から*j*にエッジがあるなら、*i*が*j*より先に現れるようにノードを並べる

- リスト・スケジューリング
  - 有限資源におけるトポロジカル・ソート
  - NP困難
  - 発見的手法
    - 優先度：ノードのグラフにおける高さ
      - そのノードの実行から、それに依存する全命令が実行を終了するまでの最小時間
    - 準最適

優先度の計算



資源予約表

サイクル	資源		
	ALU	ALU	MEM
0			
1			
2			
3			

基本アルゴリズム

READY = 先行ノードがないノードの集合;
while (READYが空でない) {
    n = READYの中で最も優先度の高いノード;
    *n*を先行制約と資源制約の両方を満たす
        最も早いスロットにスケジュールする;
    DAG(先行制約グラフ)より*n*を削除する;
    READYを更新する;
}

スケジューリング・ポリシー

- オペレーション・スケジューリング
  - 最も高い優先度の場合は、最も高く優先されてスジューリングされる
  - （やや）高いコスト
    - 資源制約を満たす、最も早いスロットを探さなければならない
- サイクル・スケジューリング
  - サイクル毎にスケジュール可能な命令をスケジュールする
  - 必ずしも優先度順でない
    - 資源を複数サイクルにわたって占有する命令がある場合、オペレーション・スケジューリングに劣る
    - 完全にパイプライン化⇒この不利はない

サイクル・スケジューリングの欠点

fadd, fmul：レイテンシ=3、非パイプライン化
サイクル0で、優先度 fadd < fmul

サイクル	
0	fadd
1	fadd
2	fadd

fmulは遅延を満たしておらず
このサイクルにはスケジュール
できない。代わりに、faddが
スケジュールされる。

fmulは遅延を満たし、
スケジュール可能。
しかし、faddに資源を
使われているので、
スケジュールできない。

サイクル・スケジューリングのアルゴリズム(1)

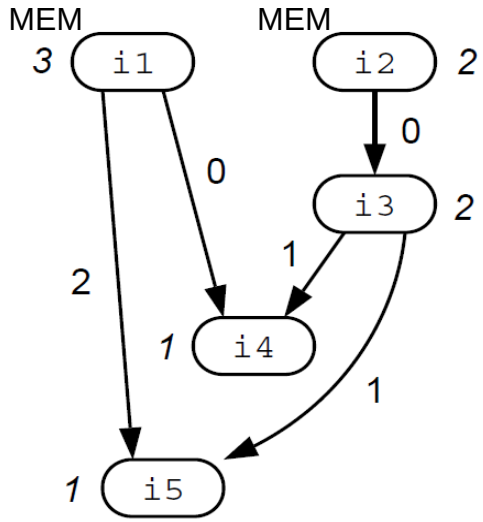
- READY集合
  - 先行ノードがないノードの集合の内、現在のサイクルにスケジュールできるノードの集合
- LEADER集合
  - 先行ノードがないノードの集合の内、現在のサイクルでは遅延の条件を満たさず、スケジュールできないノードの集合

サイクル・スケジューリングのアルゴリズム(2)

```
cycle = 0;
READY, LEADERを計算;
while (READYが空でない || LEADERが空でない）{
    foreach ノードn in READY（優先度の高い順に）{
        cycleにnのスケジュールを試みる;
        if（成功）{
            スケジュールを資源予約表に記録;
            DAGよりnを削除する;
            READYとLEADERを更新する;
        }
    }
    cycle++;
    READYとLEADERを更新する;
}
```

例(1)

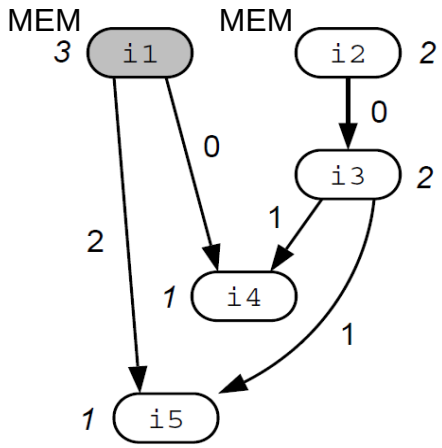
iter	cycle	ステップ	READY	LEADER	スケジュール
0	0	RとLの初期化	{i1,i2}	{}	
1	0	スケジュール	{i1,i2}	{}	(i1,0,MEM)
		ノードを削除し、RとLを更新	{i2}	{}	
	1	cycle++、RとLを更新	{i2}	{}	



cycle	ALU	MEM
0		i1

例(2)

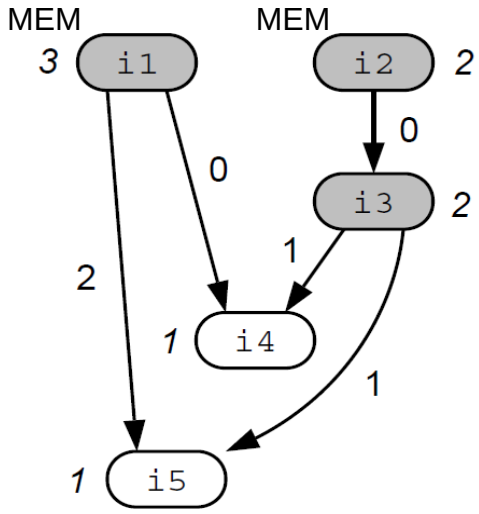
iter	cycle	ステップ	READY	LEADER	スケジュール
2	1	スケジュール	{i2}	{}	(i2,1,MEM)
		ノードを削除し、RとLを更新	{i3}	{}	
		スケジュール	{i3}	{}	(i3,1,ALU)
		ノードを削除し、RとLを更新	{}	{(i4,2), (i5,2)}	
	2	cycle++、RとLを更新	{i4,i5}	{}	



cycle	ALU	MEM
0		i1
1	i3	i2

例(3)

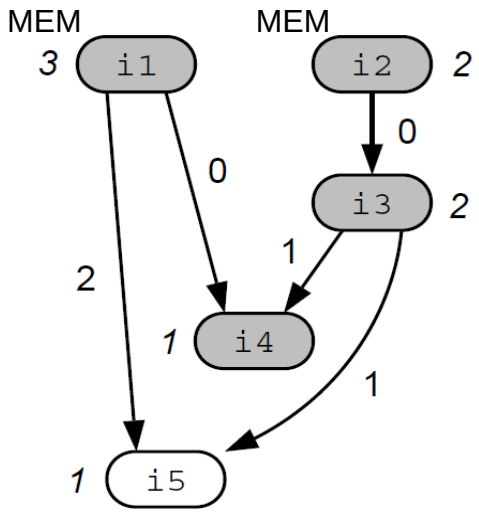
iter	cycle	ステップ	READY	LEADER	スケジュール
3	2	スケジュール	{i4,i5}	{}	(i4,2,ALU)
		ノードを削除し、RとLを更新	{i5}	{}	
	3	cycle++、RとLを更新	{i5}	{}	



cycle	ALU	MEM
0		i1
1	i3	i2
2	i4	

例(4)

iter	cycle	ステップ	READY	LEADER	スケジュール
4	3	スケジュール	{i5}	{}	(i5,3,ALU)
		ノードを削除し、RとLを更新	{}	{}	

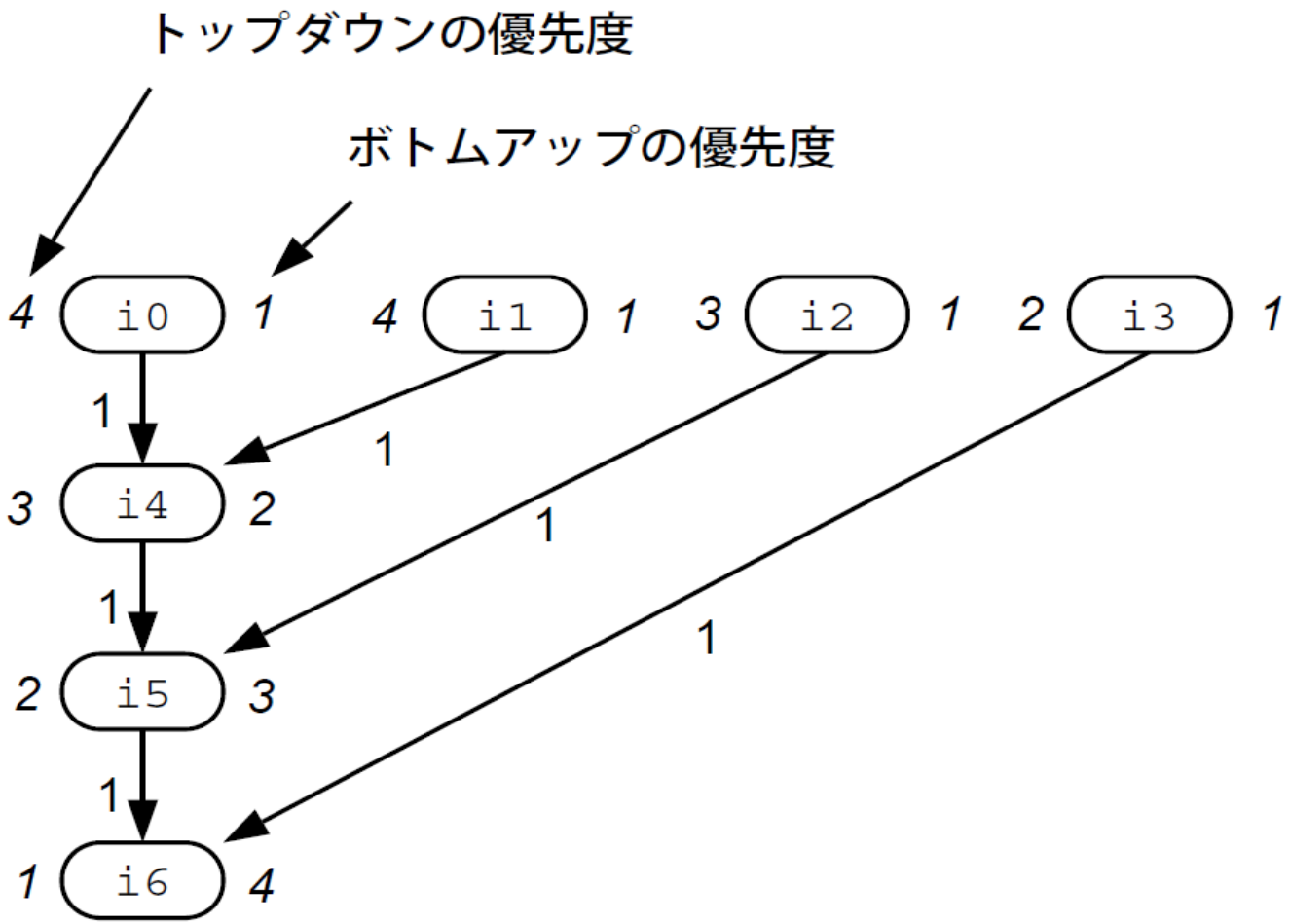


cycle	ALU	MEM
0		i1
1	i3	i2
2	i4	
3	i5	

トップダウン vs. ボトムアップ

- トップダウン
  - 根から葉
  - 「先行命令」がスケジュールされると、遅延を満たす最も「早い」サイクルにスケジュール
- ボトムアップ
  - 葉から根
  - 「後続命令」がスケジュールされると、遅延を満たす最も「遅い」サイクルにスケジュール
  - レジスタの生存期間の短縮

例


トップダウンの優先度
ボトムアップの優先度
4
i0
1
4
i1
1
3
i2
1
2
i3
1
1
1
3
i4
2
1
1
1
2
i5
3
1
1
4
i6
4

スケジューリング結果

- トップダウン

サイクル	FU1	FU2	FU2	FU3	レジスタ数
0	i0	i1	i2	i3	4
1	i4				3
2	i5				2
3	i6				1

- ボトムアップ

サイクル	FU1	FU2	FU2	FU3	レジスタ数
0	i0	i1			2
1	i4	i2			2
2	i5	i3			2
3	i6				1

少ないレジスタ

まとめ

- VLIW (Very Long Instruction Word）プロセッサ
  - 静的スケジューリング
  - 単純なハードウェア
  - 高度な命令スケジューリング

- 局所命令スケジューリング
  - リスト・スケジューリング
  - オペレーション vs. サイクル
  - トップダウン vs. ボトムアップ