Intel Developer Forum

# インテル® Itanium® 2プロセッサのプログラミング手法

**池井　満**
HPC シニア アーキテクト
インテル㈱



Intel Labs

Intel Developer Forum

# 目次

- ストリーム・ベンチのコンパイル(Fortran)
  - 最適化レポートの出力方法
  - Itanium® 2 プロセッサの演算リソース
  - ソフトウェア・パイプライニング
- 行列乗算のコンパイル (C)
  - 最適化オプション
  - 最適化ディレクティブ
- SPECintのコンパイル
  - プロシジャ間最適化 (IPO)
  - プロファイル・ガイド最適化 (PGO)



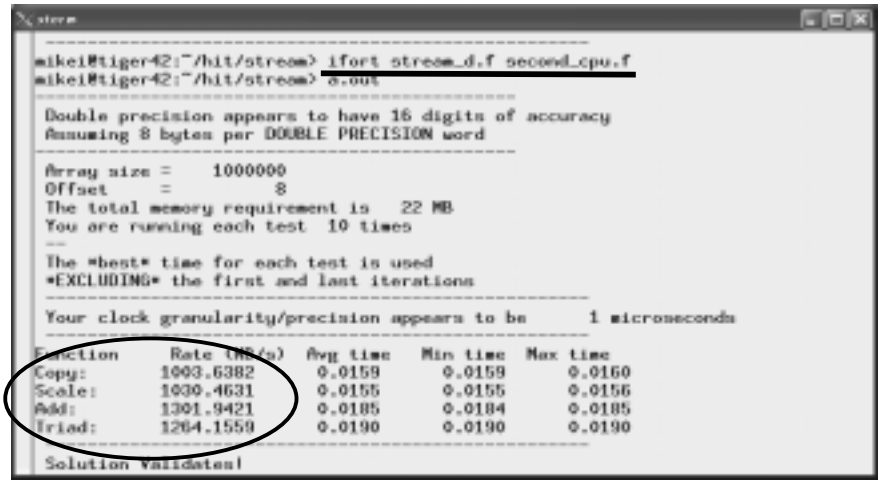
Intel Labs

# stream_d.fの実行ループ

```
171       DO 70 k = 1,ntimes
172
173           t = mysecond()
...
176           DO 30 j = 1,n
177               c(j) = a(j)
178     30    CONTINUE
…
186           DO 40 j = 1,n
187               b(j) = scalar*c(j)
188     40    CONTINUE
...
196           DO 50 j = 1,n
197               c(j) = a(j) + b(j)
198     50    CONTINUE
...
206           DO 60 j = 1,n
207               a(j) = b(j) + scalar*c(j)
208     60    CONTINUE
...
212     70 CONTINUE
```




# 単純コンパイルと実行結果



# コンパイルとリンク・オプションの確認

```
mikei@tiger42:~/hit/stream>
mikei@tiger42:~/hit/stream> ifort -dryrun stream_d.f second_cpu.f
/opt/intel_fc_80/bin/fortcom  \
    -D__INTEL_COMPILER=800 \
    -D__ELF__ \
    -D__unix__ \
    -D__unix \
    -D__linux__ \
    -D__linux \
    -D__gnu_linux__ \
    -Dunix \
    -Dlinux \
    -D__ia64 \
    -D__ia64__ \
    -Dia64 \
    -mGLOB_pack_sort_init_list \
    -I, \
    -I/opt/intel_fc_80/include \
    -I/opt/intel_fc_80/include \
    -I/opt/intel_fc_80/mubatitute_headers \
    -I/usr/include \
    -I/usr/local/electron/include64 \
    -I/usr/include \
    -O2 \
    -mP1OPT_version=802 \
    -mGLOB_source_language=GLOB_SOURCE_LANGUAGE_F90 \
    -mGLOB_tune_for_fort \
    -mGLOB_use_fort_dope_vector \
    -mP2OPT_static_promotion \
    -mP1OPT_print_version=FALSE \
    "-mGLOB_options_string=-I /opt/intel_fc_80/include -dryrun" \
    -mGLOB_cxx_limited_range=FALSE \
```




# 最適化(-O2)の内容確認

```
mikei@tiger42:~/hit/stream>
mikei@tiger42:~/hit/stream>
mikei@tiger42:~/hit/stream> ifort stream_d.f second_cpu.f -opt_report 2> rep
mikei@tiger42:~/hit/stream>
mikei@tiger42:~/hit/stream> ifort stream_d.f -c -opt_report 2> rep2
mikei@tiger42:~/hit/stream> ifort second_cpu.f -c -opt_report 2>>rep2
mikei@tiger42:~/hit/stream>
mikei@tiger42:~/hit/stream> diff rep rep2
3c3
< SWP REPORT LOG OPENED ON Fri Nov 28 15:54:53 2003
---
> SWP REPORT LOG OPENED ON Fri Nov 28 15:55:18 2003
854c854
< SWP REPORT LOG OPENED ON Fri Nov 28 15:54:53 2003
---
> SWP REPORT LOG OPENED ON Fri Nov 28 15:55:34 2003
mikei@tiger42:~/hit/stream> head rep

================================================================================
SWP REPORT LOG OPENED ON Fri Nov 28 15:54:53 2003
================================================================================

Optimization Report for: MAIN__()
Phase : Lowering
Counts:
    TOTAL transformations             :      0
```

# 最適化のレポート・オプション

- **関数名の指定**
  –opt_report_routine<name>
- **レポート・ファイル名の指定**
  –opt_report_file<name>
- **最適化フェーズの指定**
  -opt_report_phase<name>
- **最適化レポートのレベル**
  -opt_report_level[min|med|max]




# 最適化のレポート

-opt_report_phase<*phase*>

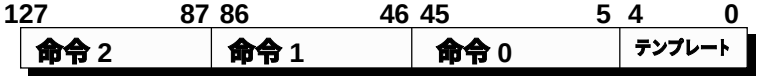
| 最適化論理名 | 最適化の内容 | 関連する最適化オプション等 |
|---|---|---|
| ipo | Interprocedural Optimizer | -ipo, -ip |
| hlo | High-level Language Optimizer | -O3 ( Loop unrolling ) |
| ilo | Intermediate Language Scalar Optimizer | |
| ecg | Itanium Compiler Code Generator | ( Software Pipelining ) |
| pgo | Profile Guided Optimizer | -prof_gen, -prof_use |
| all | All optimizers | |

# stream_d.f メインプログラム

```
Code emission report for function MAIN__ (1) in file stream_d.f
Caveat: All dynamic data based on static profiles
Code size:
        Static bundle count                                  = 519
        Static instruction count (excluding nops)            = 958
        Static instruction count (Pre) (excluding nops)      = 0 (0.0%)
        Static instruction count (SWP) (excluding nops)      = 41 (4.3%)
        Static instruction count (GCS) (excluding nops)      = 6 (0.6%)
        Static instruction count (Post) (excluding nops)     = 911 (95.1%)
        Static instruction count (Unknown) (excluding nops) = 0 (0.0%)
        Static nop count                                     = 594
        Static nop count compared to total instructions      = 38.3%
        Dynamic bundle count                                 = 6.700086e+07
        Dynamic instruction count (excluding nops)           = 1.620015e+08
        Dynamic nop count                                    = 3.900104e+07
        Dynamic nop count compared to dynamic instructions = 19.4%
        Dynamic hot instruction count (excluding nops)       = 0.000000e+00
        Dynamic cold instruction count (excluding nops)      = 0.000000e+00
Estimated exec time, in cycles                               = 4.680022e+08
Estimated exec time, in cycles (Pre)                  = 0.000000e+00 (0.0%)
Estimated exec time, in cycles (SWP)                  = 4.659997e+08 (99.6%)
Estimated exec time, in cycles (GCS)                  = 2.000008e+06 (0.4%)
Estimated exec time, in cycles (Post)                 = 2.503679e+03 (0.0%)
Estimated exec time, in cycles (Unknown)              = 0.000000e+00 (0.0%)
Average IPC = 0.35
"rep" 986L, 45587C                                          198,1-8       18%
```

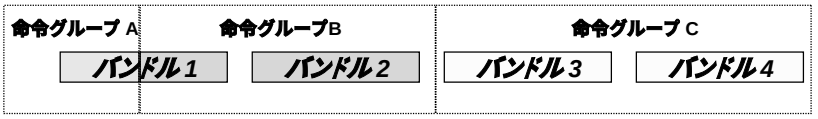



# 命令バンドル




*バンドルフォーマット*

- バンドル
  - 3つの命令スロット (各41-ビット)とテンプレートフィールド(5ビット)
  - 命令グループは幾つかのバンドルにまたがることができる
  - *例:*




**柔軟性のある発行と並列実行**

# 命令グループ

- 同時に実行可能な1つ以上の命令の固まり（命令数の制限無し）
  - お互いに依存性のない隣接する命令は並列して実行することができる
- ブレーク・コード（;;）をアセンブリ・コード中に記述することによって、命令グループの境界を指示する、もしくは実行時に分岐によってターミネートする
- ***例:***

```
add     r31 = r5, r6 ;;        Instr Group A
mov     r4 = r31               Instr Group B
add     r2 = r8, r9 ;;
mov     r7 = r2                Instr Group C
mov     r15 = r27
mov     r16 = r28
mov     r17 = r29
add     r3 = r11, r20 ;;
mov     r10 = r3               Instr Group D
```




# stream_d.f メインプログラム

```
Code emission report for function MAIN__ (1) in file stream_d.f
Caveat: All dynamic data based on static profiles
Code size:
        Static bundle count                                  = 519
        Static instruction count (excluding nops)            = 958
        Static instruction count (Pre) (excluding nops)      = 0 (0.0%)
        Static instruction count (SWP) (excluding nops)      = 41 (4.3%)
        Static instruction count (GCS) (excluding nops)      = 6 (0.6%)
        Static instruction count (Post) (excluding nops)     = 911 (95.1%)
        Static instruction count (Unknown) (excluding nops) = 0 (0.0%)
        Static nop count                                     = 594
        Static nop count compared to total instructions      = 38.3%
        Dynamic bundle count                                 = 6.700086e+07
        Dynamic instruction count (excluding nops)           = 1.620015e+08
        Dynamic nop count                                    = 3.900104e+07
        Dynamic nop count compared to dynamic instructions = 19.4%
        Dynamic hot instruction count (excluding nops)       = 0.000000e+00
        Dynamic cold instruction count (excluding nops)      = 0.000000e+00
Estimated exec time, in cycles                               = 4.680022e+08
Estimated exec time, in cycles (Pre)                  = 0.000000e+00 (0.0%)
Estimated exec time, in cycles (SWP)                  = 4.659997e+08 (99.6%)
Estimated exec time, in cycles (GCS)                  = 2.000000e+06 (0.4%)
Estimated exec time, in cycles (Post)                 = 2.503679e+03 (0.0%)
Estimated exec time, in cycles (Unknown)              = 0.000000e+00 (0.0%)
Average IPC = 0.35
 rep  986L, 45587C                                          198,1-8       18%
```

# ソフトウエア・パイプライン

Intel Developer Forum

- 考察

*C コード*:
```
for (i = 0; i < n; i++)
        y[i] = a * x[i];
```

*擬似コード*:
```
loop:
        load xi
        fmul yi = a, xi
        store yi
        branch loop
```

4サイクル

2サイクル

- 仮定
  - 命令のレーテンシー:
    - ✓load 4 サイクル*
    - ✓fmul 2 サイクル*
    - ✓store 1 サイクル*
    - ✓branch 1 サイクル*

* ここでのサイクルカウントは実際の動作とは異なります。

  - Load, fmul, store そして branch は同じ命令グループで発行可能



Intel Labs

# ソフトウエア・パイプライン

Intel Developer Forum

*For n = 8*

| | | | |
|---|---|---|---|
| *Cycle 1:* | load x1 | | |
| *Cycle 2:* | load x2 | | |
| *Cycle 3:* | load x3 | | |
| *Cycle 4*: | load x4 | | |
| *Cycle 5:* | load x5 | fmul y1=a,x1 | |
| *Cycle 6:* | load x6 | fmul y2=a,x2 | |
| *Cycle 7:* | load x7 | fmul y3=a,x3 | store y1 |
| *Cycle 8:* | load x8 | fmul y4=a,x4 | store y2 |
| *Cycle 9:* | | fmul y5=a,x5 | store y3 |
| *Cycle 10:* | | fmul y6=a,x6 | store y4 |
| *Cycle 11:* | | fmul y7=a,x7 | store y5 |
| *Cycle 12:* | | fmul y8=a,x8 | store y6 |
| *Cycle 13:* | | | store y7 |
| *Cycle 14:* | | | store y8 |

プロローグ

カーネル

エピローグ

* ここでのサイクルカウントは実際の動作とは異なります

**この例では1回あたりのループに7サイクル必要**



Intel Labs

# stream_d.fの207行ループ

```
 nop.i   0 ;;
// Block 42: lentry lexit ltail collapsed pipelined  Pred: 41 42     Succ:
   42 43
  -S
// Freq 9.0e+06
 }
.b1_42:
 {   .mfi
  (p16) ldfd    f32=[r18],8                              //0:207  395
  (p27) fma.d   f56=f6,f43,f55                           //11:207  397
        nop.i   0
 }
 {   .mmb
  (p16) ldfd    f44=[r17],8                              //0:207  396
  (p31) stfd    [r16]=f60,8                              //15:207  398
// Branch taken probability 0.99
        br.ctop.sptk    .b1_42 ;;                        //0:206  402
// Block 43: epilog  Pred: 42     Succ: 44  -O
// Freq 9.0e+00
 }
```




# stream_d.fの207行ループ

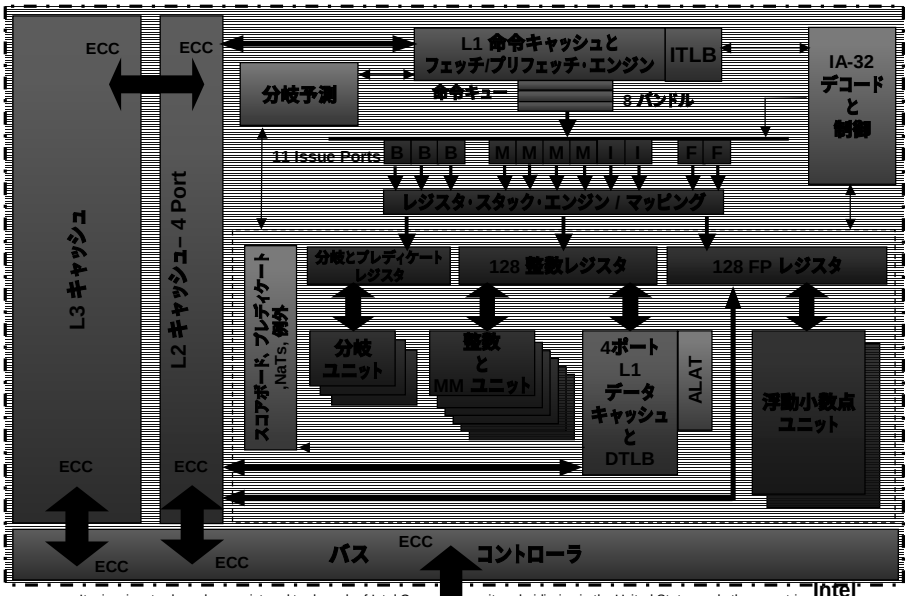
```
================================================================================
SWP REPORT LOG OPENED ON Sun Nov 30 16:39:51 2003
================================================================================
...
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Swp report for loop at line 207 in MAIN__ in file stream_d.f

        Resource II   =        1
        Recurrence II =        0
        Minimum II    =        1
        Scheduled II  =        1

        Percent of Resource II needed by arithmetic ops     = 100%
        Percent of Resource II needed by memory ops         = 100%
        Percent of Resource II needed by floating point ops = 100%

        Number of stages in the software pipeline =    16
--------------------------------------------------------------------------------
```

# Initiation Interval (II)

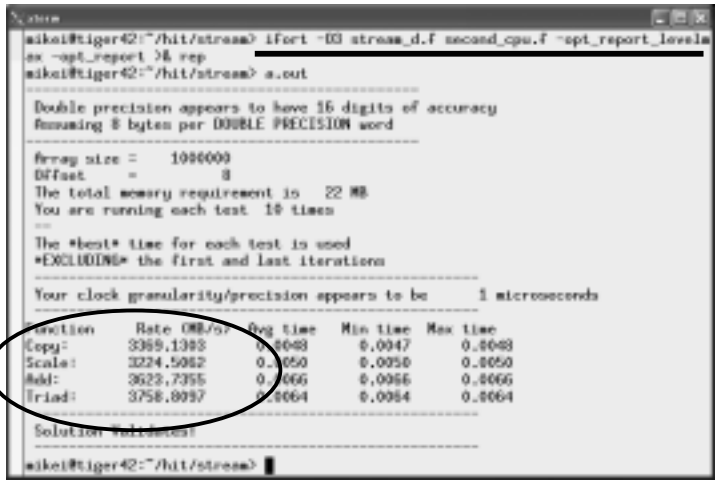
- **ループの繰り返しの開始間隔サイクル**
  - **ループ内の処理を実行するために必要な最小サイクル数**
- **リソースII**
  - **プロセッサの演算リソースの制限によるII**
- **再帰 (リカーランス) II**
  - **ループ内のデータ依存性により必要なII**
- **最小II**
  - MAX (**リソースII, 再帰II**) : **必要最小II**
- **スケジュールII**
  - **実際にコンパイラが適用したII**




## Intel® Itanium® 2プロセッサのブロック・ダイアグラム



# -O3コンパイルと実行結果






# プリフェッチとロードペア

```
================================================================================
High Level Optimizer Report for: MAIN__
...
Estimate of max_trip_count of loop at line 206=125001
Total #of lines prefetched in MAIN__ for loop in line 206=3
   # of spatial prefetches in MAIN__ for loop in line 206=3, dist=100
#
...
Load-pair formed at line 197 , 197   [Method = Aligned]
Load-pair formed at line 197 , 197   [Method = Aligned]
Load-pair formed at line 207 , 207   [Method = Aligned]
Load-pair formed at line 207 , 207   [Method = Aligned]
Load-pair formed at line 207 , 207   [Method = Aligned]
Load-pair formed at line 207 , 207   [Method = Aligned]
Load-pair formed at line 207 , 207   [Method = Aligned]
Load-pair formed at line 207 , 207   [Method = Aligned]
Load-pair formed at line 207 , 207   [Method = Aligned]
Load-pair formed at line 207 , 207   [Method = Aligned]
Load-pair formed at line 215 , 215   [Method = Aligned]
```

# プリフェッチとロードペア

```
Block, Unroll, Jam Report:
(loop line numbers, unroll factors and type of transformation)
Loop at line 151 unrolled without remainder by 4
Loop at line 158 unrolled without remainder by 8
Loop at line 176 unrolled without remainder by 4
Loop at line 186 unrolled without remainder by 4
Loop at line 196 unrolled without remainder by 4
Loop at line 206 unrolled without remainder by 8
Loop at line 216 completely unrolled by 4
...
--------------------------------------------------------------------------------
Swp report for loop at line 207 in MAIN__ in file stream_d.f

        Resource II   =        6
        Recurrence II =        0
        Minimum II    =        6
        Scheduled II  =        7

        Percent of Resource II needed by arithmetic ops     =  83%
        Percent of Resource II needed by memory ops         =  83%
        Percent of Resource II needed by floating point ops =  67%

        Number of stages in the software pipeline =    3
--------------------------------------------------------------------------------
```




# ループ・アンローリング

```
...
206           DO 60 j = 1,n
207               a(j) = b(j) + scalar*c(j)
208    60     CONTINUE
...


...
206           DO jj = 1,n/8
207               j=(jj-1)*8
207               a(j) = b(j) + scalar*c(j)
207               a(j+1) = b(j+1) + scalar*c(j+1)
207               a(j+2) = b(j+2) + scalar*c(j+2)
207               a(j+3) = b(j+3) + scalar*c(j+3)
207               a(j+4) = b(j+4) + scalar*c(j+4)
207               a(j+5) = b(j+5) + scalar*c(j+5)
207               a(j+6) = b(j+6) + scalar*c(j+6)
207               a(j+7) = b(j+7) + scalar*c(j+7)
207           END DO
207           DO 60 j = (n/8)*8, n+mod(n,8)
207               a(j) = b(j) + scalar*c(j)
208    60     CONTINUE
...
```

# プリフェッチとロードペア

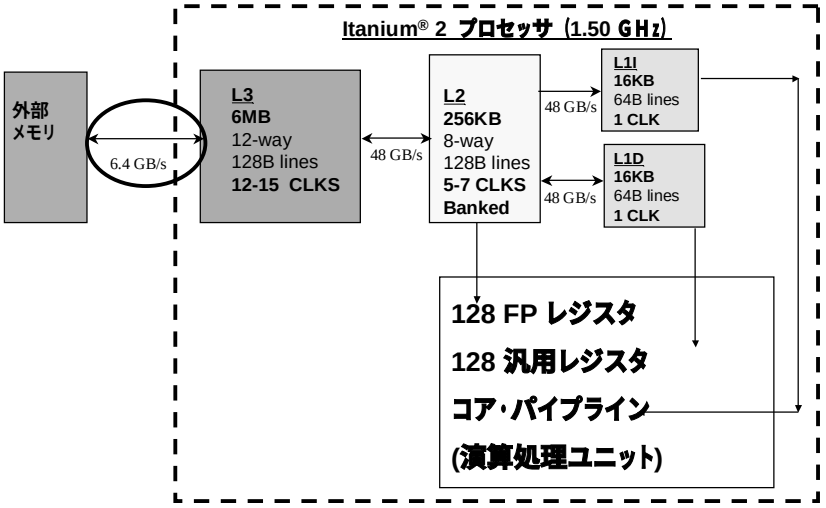
```
// Block 44: lentry lexit ltail collapsed pipelined  Pred: 44 43     Succ: 44
    45
  -S
// Freq 2.7e+07
 }
.b1_44:
 {   .mfi
  (p16) ldfpd   f41,f32=[r39]                          //0:207  574
  (p18) fma.d   f55=f6,f72,f45                         //14:207  594
        nop.i   0
 }
 {   .mfi
  (p16) ldfpd   f53,f50=[r38]                          //0:207  575
  (p18) fma.d   f56=f6,f69,f36                         //14:207  596
        nop.i   0 ;;
 }
 {   .mmi
  (p18) ldfpd   f45,f38=[r42]                          //1:207  588
  (p18) ldfpd   f60,f57=[r44]                          //1:207  581
        nop.i   0
 }
 {   .mmi
  (p18) stfd    [r29]=f39,64                           //15:207  577
  (p18) stfd    [r28]=f48,64                           //15:207  579
        nop.i   0 ;;
 }
 {   .mmi
  (p16) ldfpd   f48,f39=[r46]                          //2:207  586
  (p16) ldfpd   f65,f62=[r37]                          //2:207  587
        nop.i   0
 }
 {   .mmi
  (p18) stfd    [r27]=f34,64                           //16:207  583
  (p18) stfd    [r26]=f43,64                           //16:207  585
        nop.i   0 ;;
 }
 {   .mmi
  (p18) ldfpd   f70,f67=[r41]                          //3:207  593
  (p18) ldfpd   f43,f34=[r35]                          //3:207  592
  (p18) add     r35=64,r35                             //3.206  618
 }
 {   .mmi
  (p18) stfd    [r25]=f47,64                           //17:207  591
  (p18) stfd    [r24]=f38,64                           //17.207  588
  (p18) add     r40=64,r41 ;;                          //3.206  619
 }
```

```
{   .mmf
  (p16) add     r36=64,r37                             //4:206  617
  (p16) lfetch.nt1      [r34]                          //4:206  598
  (p17) fma.d   f47=f6,f51,f33                         //11:207  578
 }
 {   .mmf
  (p18) stfd    [r23]=f56,64                           //18:207  597
  (p18) stfd    [r22]=f55,64                           //18:207  595
  (p17) fma.d   f38=f6,f54,f42 ;;                      //11:207  576
 }
 {   .mmf
  (p18) add     r37=64,r38                             //5:206  613
  (p18) add     r41=64,r42                             //5.206  614
  (p17) fma.d   f33=f6,f51,f46                         //13:207  582
 }
 {   .mmi
  (p18) add     r43=64,r44                             //5.206  615
  (p16) add     r45=64,r46                             //5.206  616
  (p17) fma.d   f42=f6,f50,f37 ;;                      //12:207  584
 }
 {   .mfi
  (p16) lfetch.nt1      [r21],64                       //6:206  602
  (p17) fma.d   f37=f6,f66,f49                         //13:207  588
  (p16) add     r32=128,r34                            //6:206  599
 }
 {   .mfb
  (p16) add     r38=64,r39                             //6:206  612
  (p17) fma.d   f46=f6,f63,f40                         //13:207  590
// Branch taken probability 1.00
        br.ctop.sptk    .b1_44 ;;                      //6:206  620
// Block 45: epilog  Pred: 44     Succ: 46  -O
```



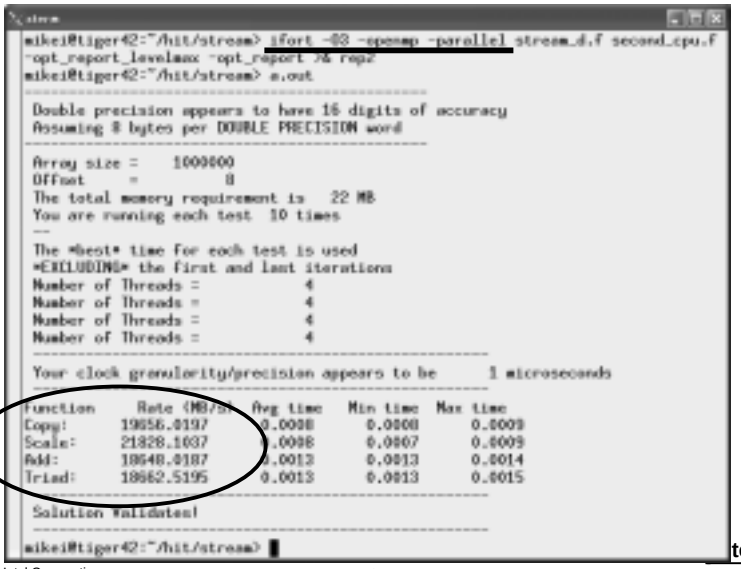


# Itanium® 2 プロセッサ メモリ 構造



# -openmp 並列化コンパイルと実行結果

Intel Developer Forum




Intel Labs





Intel Developer Forum

# 目次

- ストリーム・ベンチのコンパイル(Fortran)
  - 最適化レポートの出力方法
  - Itanium® 2 プロセッサの演算リソース
  - ソフトウェア・パイプライニング
- 行列乗算のコンパイル (C)
  - 最適化オプション
  - 最適化ディレクティブ
- SPECintのコンパイル
  - プロシジャ間最適化 (IPO)
  - プロファイル・ガイド最適化 (PGO)

Intel Labs

# multiply_d.c プログラム

Intel Developer Forum

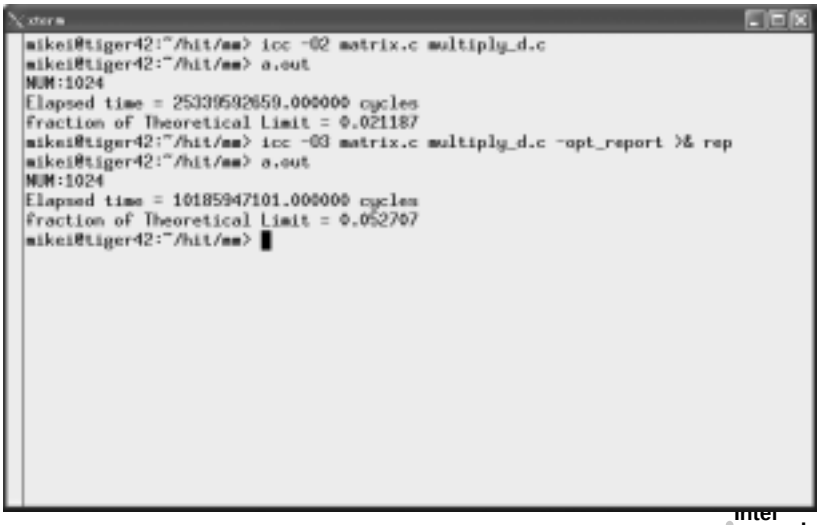
```
 1 #include "multiply_d.h"
 2
 3 // matrix multiply routine
 4
 5 void multiply_d(double a[][DIM], double  b[][DIM], double  c[][DIM])
 6 {
 7         int i,j,k;
 8         double temp;
 9         for(i=0;i<NUM;i++) {
10             for(k=0;k<NUM;k++) {
11                 for(j=0;j<NUM;j++) {
12                         c[i][j] = c[i][j] + a[i][k] * b[k][j];
13                 }
14             }
15         }
16 }
17
```



Intel Labs

# 単純コンパイルと実行結果

Intel Developer Forum






Intel Labs

# multiply_d.cの12行ループ

Intel Developer Forum

```
--------------------------------------------------------------------------------
Swp report for loop at line 12 in multiply_d in file multiply_d.c

        Resource II   =        4
        Recurrence II =        15
        Minimum II    =        15
        Scheduled II  =        18

        Percent of Resource II needed by arithmetic ops     = 100%
        Percent of Resource II needed by memory ops         = 100%
        Percent of Resource II needed by floating point ops =  25%

        Number of stages in the software pipeline =    2

        Following are the loop-carried memory dependence edges:
        Store  at line    12 --> Load   at line    12
        Store  at line    12 --> Load   at line    12
        Store  at line    12 --> Load   at line    12
        Store  at line    12 --> Load   at line    12
        Load   at line    12 --> Store  at line    12
        Load   at line    12 --> Store  at line    12
        Store  at line    12 --> Load   at line    12
        Store  at line    12 --> Load   at line    12
--------------------------------------------------------------------------------
```





Intel Labs

# -fno_alias による依存性解消

Intel Developer Forum

```
mikei@tiger42:~/hit/mm> icc -O2 matrix.c multiply_d.c
mikei@tiger42:~/hit/mm> a.out
NUM:1024
Elapsed time = 25339592659.000000 cycles
Fraction of Theoretical Limit = 0.021187
mikei@tiger42:~/hit/mm> icc -O3 matrix.c multiply_d.c -opt_report >& rep
mikei@tiger42:~/hit/mm> a.out
NUM:1024
Elapsed time = 10185947101.000000 cycles
Fraction of Theoretical Limit = 0.052707
mikei@tiger42:~/hit/mm> icc -O3 -fno-alias matrix.c multiply_d.c -opt_report >&
rep
mikei@tiger42:~/hit/mm> a.out
NUM:1024
Elapsed time = 2293155112.000000 cycles
Fraction of Theoretical Limit = 0.234119
mikei@tiger42:~/hit/mm>
```





Intel Labs

# multiply_d.cの12行ループ

Intel Developer Forum

- ----------
- Swp report for loop at line 12 in multiply_d in file multiply_d.c
- Resource II = 3
- Recurrence II = 0
- Minimum II = 3
- Scheduled II = 3
- Percent of Resource II needed by arithmetic ops = 100%
- Percent of Resource II needed by memory ops = 100%
- Percent of Resource II needed by floating point ops = 33%
- Number of stages in the software pipeline = 6
- ----------



Intel Labs

# #pragma ivdep の挿入

Intel Developer Forum

```
 1 #include "multiply_d.h"
 2
 3 // matrix multiply routine
 4
 5 void multiply_d(double a[][DIM], double  b[][DIM], double  c[][DIM])
 6 {
 7         int i,j,k;
 8         double temp;
 9         for(i=0;i<NUM;i++) {
10             for(k=0;k<NUM;k++) {
11 #pragma ivdep
12                 for(j=0;j<NUM;j++) {
13                         c[i][j] = c[i][j] + a[i][k] * b[k][j];
14                 }
15             }
16         }
17 }
18
```



Intel Labs

# ivdep による依存性解消





# 最適化のディレクティブ

| C言語シンタックス | Fortranシンタックス |
|---|---|
| #pragma [no]swp | !DEC$ [NO]SWP |
| #pragma loop count (10000) | !DEC$ LOOP COUNT (n) |
| #pragma distribute point | !DEC$ DISTRIBUTE POINT |
| #pragma [no]unroll(n) | !DEC$ [NO]UNROLL(n) |
| #pragma [no]prefetch a,b | !DEC$ [NO]PREFETCH |
| #pragma ivdep | !DEC$ IVDEP |

# ディレクティブの使用例

Intel Developer Forum

```
#pragma swp
for (i=0; i<m ; i++)
{
  if (a[i]==0)
  {
     b[i]=a[i]+1;
  }
  else
  {
     b[i]=a[i]*2;
  }
}
```

```
!DEC$ IVDEP
   do j=1,n
      a(b(j)) = a(b(j))+1
   enddo
```



Intel Labs

# プロシジャ間 (IPO) 最適化

Intel Developer Forum

```
mikei@tiger42:~/hit/mm> icc -O2 matrix.c multiply_d.c
mikei@tiger42:~/hit/mm> a.out
NUM:1024
Elapsed time = 25339276033.000000 cycles
fraction of Theoretical Limit = 0.021187
mikei@tiger42:~/hit/mm> icc -O3 matrix.c multiply_d.c -opt_report >& rep
mikei@tiger42:~/hit/mm> a.out
NUM:1024
Elapsed time = 10185434135.000000 cycles
fraction of Theoretical Limit = 0.052710
mikei@tiger42:~/hit/mm> icc -O3 -fno-alias matrix.c multiply_d.c -opt_report >&
rep
mikei@tiger42:~/hit/mm> a.out
NUM:1024
Elapsed time = 2293148869.000000 cycles
fraction of Theoretical Limit = 0.234120
mikei@tiger42:~/hit/mm> icc -O3 -fno-alias -ipo matrix.c multiply_d.c -opt_repor
t >& rep
mikei@tiger42:~/hit/mm> a.out
NUM:1024
Elapsed time = 1074861459.000000 cycles
fraction of Theoretical Limit = 0.499479
mikei@tiger42:~/hit/mm>
```



Intel Labs

Intel Developer Forum

# 目次

- ストリーム・ベンチのコンパイル(Fortran)
  - 最適化レポートの出力方法
  - Itanium® 2 プロセッサの演算リソース
  - ソフトウェア・パイプライニング
- 行列乗算のコンパイル (C)
  - 最適化オプション
  - 最適化ディレクティブ
- SPECintのコンパイル
  - プロシジャ間最適化 (IPO)
  - プロファイル・ガイド最適化 (PGO)



Intel Labs

Intel Developer Forum

# Spec*2000 から gzipの性能

- gcc のコンパイル実行結果 185.151s
- ecc 45.146s (4.10x)
- -O3 オプション 44.726s (4.14x)
- -O3 –ipo 43.341s (4.27x)
- -O3 –prof_gen ….
- -O3 –ipo –prof_use **34.305 (5.40x)**

Linux kernel 2.4.19-smp,Intel®C/C++ compiler 7.1 #20030703,gcc 2.96
Tiger4 system Intel® Itanium 2 プロセッサ1.5GHz6MB Cache 2-way



Intel Labs

# gcc によるコンパイル実行

Intel Developer Forum

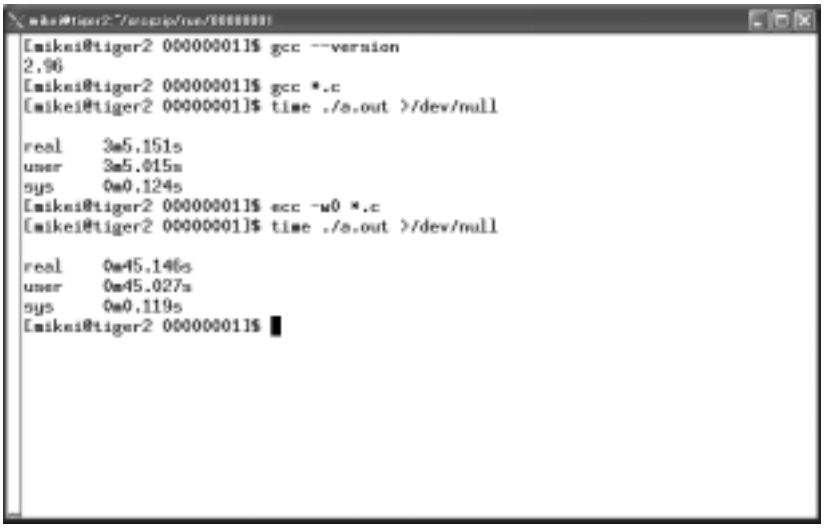





Intel Labs

# eccによるコンパイル実行

Intel Developer Forum








Intel Labs

# -O3によるコンパイル実行

Intel Developer Forum

```
[mikei@tiger2 00000001]$ gcc --version
2.96
[mikei@tiger2 00000001]$ gcc *.c
[mikei@tiger2 00000001]$ time ./a.out >/dev/null

real    3m5.151s
user    3m5.015s
sys     0m0.124s
[mikei@tiger2 00000001]$ ecc -w0 *.c
[mikei@tiger2 00000001]$ time ./a.out >/dev/null

real    0m45.146s
user    0m45.027s
sys     0m0.119s
[mikei@tiger2 00000001]$ ecc -O3 -w0 *.c
[mikei@tiger2 00000001]$ time ./a.out >/dev/null

real    0m44.726s
user    0m44.604s
sys     0m0.125s
[mikei@tiger2 00000001]$
```



Intel Labs

# -O3によるコンパイル実行( ls )

Intel Developer Forum

```
2.96
[mikei@tiger2 00000001]$ gcc *.c
[mikei@tiger2 00000001]$ time ./a.out >/dev/null

real    3m5.151s
user    3m5.015s
sys     0m0.124s
[mikei@tiger2 00000001]$ ecc -w0 *.c
[mikei@tiger2 00000001]$ time ./a.out >/dev/null

real    0m45.146s
user    0m45.027s
sys     0m0.119s
[mikei@tiger2 00000001]$ ecc -O3 -w0 *.c
[mikei@tiger2 00000001]$ time ./a.out >/dev/null

real    0m44.726s
user    0m44.604s
sys     0m0.125s
[mikei@tiger2 00000001]$ ls *.[ch]
bits.c     getopt.c  gzip.h     lzw.h       tailor.h  unlzw.c   util.c
crypt.h    getopt.h  inflate.c  revision.h  trees.c   unpack.c  zip.c
deflate.c  gzip.c    lzw.c      spec.c      unlzh.c   unzip.c
[mikei@tiger2 00000001]$
```



Intel Labs

# -ipoによるコンパイル実行

Intel Developer Forum

```
crypt.h    getopt.h  inflate.c  revision.h  trees.c   unpack.c  zip.c
deflate.c  gzip.c    lzw.c      spec.c      unlzh.c   unzip.c
[mikei@tiger2 00000001]$ ecc -O3 -ipo -w0 *.c
IPO: using IR for /tmp/eccdbfc9H.o
IPO: using IR for /tmp/eccSPrlHM.o
IPO: using IR for /tmp/ecciAdufR.o
IPO: using IR for /tmp/eccr61CNV.o
IPO: using IR for /tmp/eccpkYL10.o
IPO: using IR for /tmp/eccC5RUT4.o
IPO: using IR for /tmp/eccHI03r9.o
IPO: using IR for /tmp/eccbW5c0d.o
IPO: using IR for /tmp/eccm20lyi.o
IPO: using IR for /tmp/eccp67u6m.o
IPO: using IR for /tmp/ecc0riEEr.o
IPO: using IR for /tmp/eccWQzNcw.o
IPO: using IR for /tmp/eccZyUWKA.o
IPO: using IR for /tmp/eccqKc6iF.o
IPO: performing multi-file optimizations
[mikei@tiger2 00000001]$ time ./a.out >/dev/null

real    0m43.341s
user    0m43.216s
sys     0m0.127s
[mikei@tiger2 00000001]$
```



Intel Labs

# -prof_genによるコンパイル実行

Intel Developer Forum

```
[mikei@tiger2 00000001]$ ecc -O3  -prof_gen -w0 *.c
[mikei@tiger2 00000001]$ time ./a.out >/dev/null

real    2m32.316s
user    2m32.190s
sys     0m0.122s
[mikei@tiger2 00000001]$ ecc -O3 -ipo -prof_use -w0 *.c
/opt/intel/compiler70/ia64/bin/profmerge: WARNING: existing ./pgopti.dpi will be
 overwritten.
/opt/intel/compiler70/ia64/bin/profmerge: merging dynamic file: 3f717dfd_11203.d
yn
WARNING: gzip.c, total routines: 3, routines w/profile info: 1
WARNING: inflate.c, total routines: 8, routines w/profile info: 7
WARNING: lzw.c, total routines: 1, routines w/profile info: 0
WARNING: spec.c, total routines: 16, routines w/profile info: 11
WARNING: trees.c, total routines: 14, routines w/profile info: 13
WARNING: unlzh.c, total routines: 12, routines w/profile info: 0
WARNING: unlzw.c, total routines: 1, routines w/profile info: 0
WARNING: unpack.c, total routines: 3, routines w/profile info: 0
WARNING: unzip.c, total routines: 2, routines w/profile info: 1
WARNING: util.c, total routines: 17, routines w/profile info: 6
IPO: using IR for /tmp/ecc3gdL9Y.o
IPO: using IR for /tmp/eccAjrWDs.o
IPO: using IR for /tmp/eccLM77V.o
```



Intel Labs

# -prof_useによるコンパイル実行

Intel Developer Forum

```
WARNING: unpack.c, total routines: 3, routines w/profile info: 0
WARNING: unzip.c, total routines: 2, routines w/profile info: 1
WARNING: util.c, total routines: 17, routines w/profile info: 6
IPO: using IR for /tmp/ecc3gdL9Y.o
IPO: using IR for /tmp/eccAjrWDa.o
IPO: using IR for /tmp/ecclLM77V.o
IPO: using IR for /tmp/eccgtbjCp.o
IPO: using IR for /tmp/eccluzu6S.o
IPO: using IR for /tmp/eccQA0FAm.o
IPO: using IR for /tmp/eccywzR4P.o
IPO: using IR for /tmp/ecc2ub3yj.o
IPO: using IR for /tmp/eccswMe3M.o
IPO: using IR for /tmp/ecc6rqqxg.o
IPO: using IR for /tmp/eccJncC1J.o
IPO: using IR for /tmp/eccQQ1Nvd.o
IPO: using IR for /tmp/eccknUZ2G.o
IPO: using IR for /tmp/eccleMbua.o
IPO: performing multi-file optimizations
[mikei@tiger2 00000001]$ time ./a.out >/dev/null

real    0m34.305s
user    0m34.184s
sys     0m0.122s
[mikei@tiger2 00000001]$
```

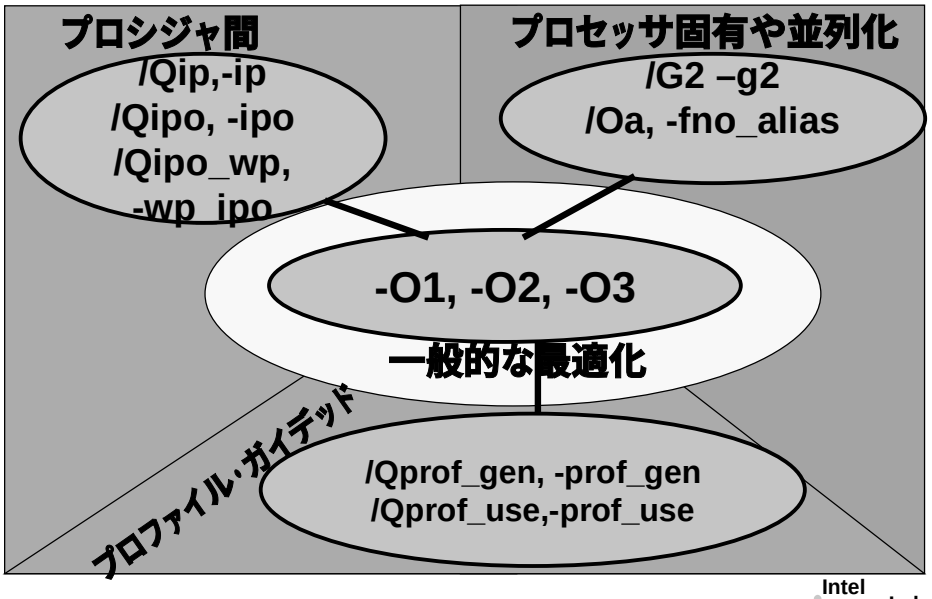


Intel Labs

最適化コンパイラ

# Performance Switches

Intel Developer Forum






Intel Labs

Intel Developer Forum

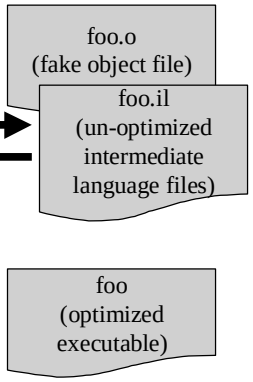
# 使用方法

1. Compile program
   `icc -c -ipo foo.c`

2. Link program
   `icc -o foo -ipo foo.o`

   2a. Compiler performs whole-program optimizations

   2b. Compiler invokes linker to produce executable






Intel Labs



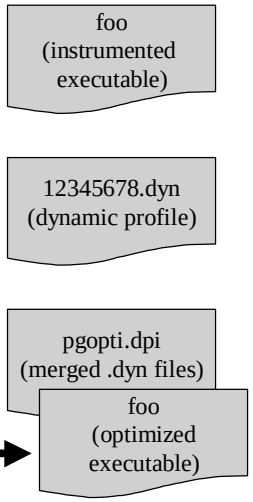
Intel Developer Forum

# 使用方法

1. Compile+link to add instrumentation
   `icc –o foo -prof_gen foo.c`

2. Execute instrumented program
   `./foo`

3. Compile+link using feedback
   `icc –o foo -prof_use foo.c`






Intel Labs

# まとめ

- コンパイラの最適化レポートを利用して重要ループのソフトウェア・パイプライン化を確認しましょう
  - プロセッサのリソースは有効に活用されていますか？
  - 偽の依存性はないでしょうか？
- ディレクティブやコンパイル時のオプションを用いてコンパイラの最適化をサポートしましょう
  - 依存性を無視したら最適化されますか？
  - コンパイラを助ける適切なデイレクティブはどれでしょうか？
  - 適用可能な場合はインテルのパフォーマンス・ライブラリを使用しましょう
- プロシジャ間最適化 (IPO) とプロファイル・ガイド最適化 (PGO) は必ず試みましょう
  - Itanium® アーキテクチャでは特に有効です




# 御参考

- 開発を用意にするインテルのパフォーマンス・ツールを利用しましょう
  - インテル® パフォーマンス コンパイラ
  - インテル® VTune™ パフォーマンス アナライザ
  - インテル® パフォーマンス ライブラリ
  - http://www.intel.co.jp/jp/developer/software/products/compilers
- 弊社のパートナXLsoft (株) により日本語のサポートが得られます
  - http://www.xlsoft.com/jp/products/vtune/perftool.htm
  - https://premier.intel.com (英語によるサポート)
- インテルSWカレッジによるトレーニング
  - http://www.intel.com/software/products/college (日本でのトレーニングについては要相談)
- パフォーマンスのチューニングやカウンタの説明には
  - IA-64 プロセッサ基本講座 池井 満著 2000年8月 オーム社
  - インテル® Itanium® 2 プロセッサ リファレンス・マニュアル ソフトウェアの開発と最適化

  http://developer.intel.com/design/itanium2/manuals/

# backup

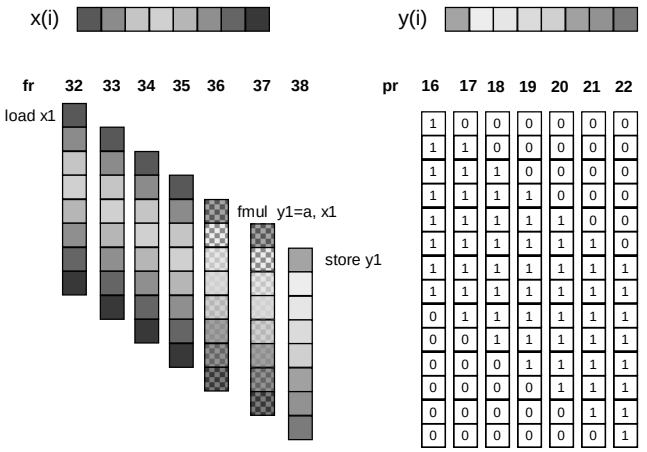



# ソフトウエア・パイプライン

- *実際のコード例*

```
// 初期化
        mov pr.rot=0                // Clear all rotating pred regs
        cmp.eq p16,p0=r0,r0         // Set p16=1
        mov ar.lc=7                 // Set loop counter to n-1
        mov ar.ec=7                 // Set epilog counter # of stages
        ...
loop:
        { .mfi
(p16)   ldfd f32=[r32],8            // Stage 1: Load x
(p20)   fmpy.d f36=f6,f36           // Stage 5: y=a*x
        nop.i 0
        }
        { .mfb
(p22)   stfd [r33]=f38,8            // Stage 7: Store y
        nop.f 0
        br.ctop.sptk.few  loop      // Branch back
        }
```

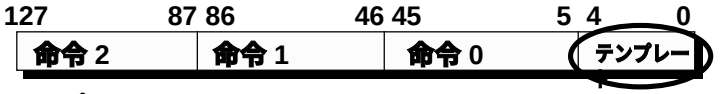
# ソフトウエア・パイプライン




*ここでのサイクルカウントは実際の動作とは異なります
注：これはアニメーションスライド

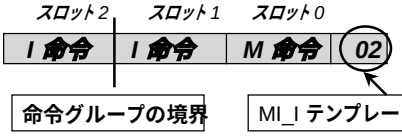



# 命令テンプレート



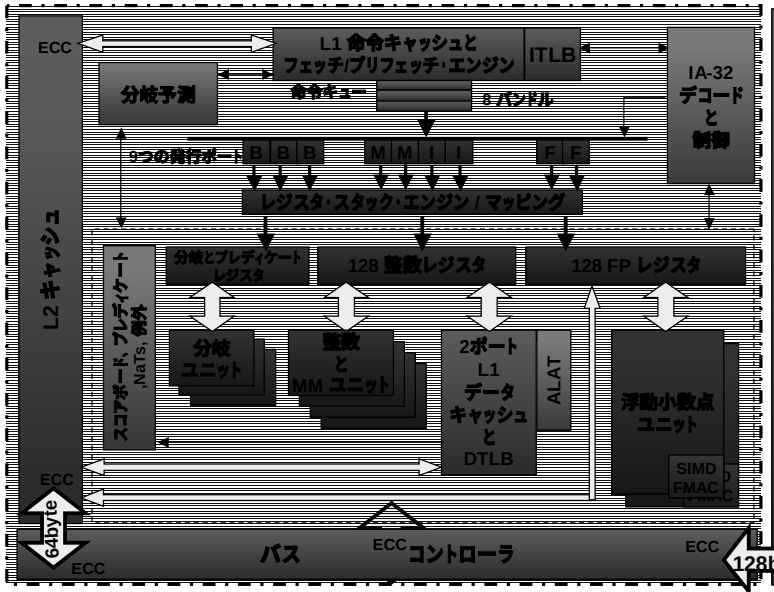

- テンプレート
  - 各命令スロットの実行ユニットタイプを指定
    - MII, MLX, MMI, MFI, MMF, MIB, MBB, BBB, MMB, MFB
  - バンドル間やバンドル内の依存性を指定
    - MI_I, M_MI
    - MII_, MLX_, MMI_, 他
  - *例:*

M= メモリ
I= 整数
A= メモリ/整数
F= 浮動小数点
L＋X= long即値
B= 分岐



# Intel® Itanium® プロセッサの ブロック・ダイアグラム

Intel Developer Forum








Labs