スーパースカラ・プロセッサの構成方式に関する研究

久我, 守弘
九州大学総合理工学研究科情報システム学専攻

https://doi.org/10.11501/3060381

出版情報：九州大学, 1991, 博士（工学）, 課程博士
バージョン：
権利関係：

第 9 章

結論

本章では，本研究の成果についてまとめ，今後の課題を明らかにする．

9.1 研究の成果

本研究では，スーパースカラ方式に基づくプロセッサの構成方式について，体系的に整理し，スーパースカラ・プロセッサの 1 実現方式を提案・評価を行った．本研究で得られた成果を以下にまとめる．

9.1.1 スーパースカラ方式の存在意義

スーパースカラ・プロセッサは，プログラムに内在する命令レベルの並列性を利用した並列処理を行うことで性能向上を図る．スーパースカラ方式は時間並列処理および空間並列処理利用するため，パイプライン方式，VLIW 方式および複数機能ユニット (MFU) プロセッサを発展させた方式とみなすことができる．スーパースカラ方式および VLIW 方式は，共に時間および空間並列性を利用するが，スーパースカラの VLIW に対する存在意義は，命令セット・アーキテクチャ上の互換性がとれること，さらには従来のパイプライン・プロセッサと全く同一の命令を利用することも可能であることに帰着する．

(1) スーパースカラの分類

3 つの基本選択肢によりスーパースカラ方式を分類した．

- ハザードの解消時点：静的 (S)vs. 動的 (D) ハザード解消
- 命令最適化時点：静的 (S)vs. 動的 (D) コード・スケジューリング

- 機能ユニットの均質性：均質型 (U)vs. 非均質型 (N) 機能ユニット

これらの基本的な分類を組み合わせることにより，**SSU, SSN, DSU, DSN, DDU, DDN** といったの 6 つの型に分類できる．付録 A に示すように，現在研究開発されているスーパースカラ・プロセッサの多くは DSN 型に分類できることがわかる．これは以下の理由によるところが大きい．

(a) 動的ハザード解消スーパースカラであるため，オブジェクト・コードの互換性を確保できる．

(b) 動的コード・スケジューリングは非常にハードウェアコストが大きいにもかかわらず，その性能に寄与する割合が小さい．

(c) 非均質構造であることから，応用分野向きに演算器構成を設定できる．また，現状では均質構造のスーパースカラは時期尚早であるといえる．

このことからも，現状では DSN 型スーパースカラが圧倒的に有力なプロセッサ構成であるといえる．

(2) DD 型スーパースカラ・プロセッサの設計および性能評価

DD 型スーパースカラ方式の試作プロセッサである “DDUS プロセッサ” の設計およびシミュレータによる性能評価を行った．

設計に際しては，スーパースカラ実現上の課題に対して以下なる対処を施すかが重大な問題である．DDUS プロセッサでは，以下のように構成上の選択肢の決定を行った．

(a) パイプライン・スライス・プロセッサ構成方式の提案：命令パイプラインを多重本並べる構成方式を採ることにより，要求される性能に見合うプロセッサを構築する．このような構成法を採る理由の 1 つとして，設計開始時点 (1987 年) においては，シングルチップ・スーパースカラを実現できるほど，VLSI 集積度が高くなかったことが挙げられる．

(b) 動的コード・スケジューリングの積極的な利用：命令間に内在する命令レベル並列性を最大限に引き出すため，動的コード・アルゴリズムを利用した．動的コード・アルゴリズムとしては Tomasulo のアルゴリズムが有名である．分岐命令を越えた out-of-order 実行を可能にするために，Tomasulo のアルゴリズムに大幅な拡張を施して，先行する複数の命令から受けるフロー依存および制御依存関係に対処可能なア

ルゴリズムを考案した．複数の命令からの依存関係を表現できるように，ビットマップを用いた多重依存関係表現法を考案し，自由度の高い動的コード・スケジューリング・アルゴリズムを実現した．

(c) 正確な割込み/分岐を reorder buffer により保証：動的コード・スケジューリングを導入すると，レジスタ内容などの更新が out-of-order になる可能性があり，不正確なマシン状態が発生する恐れがある．正確な割り込みおよび分岐を保証するために，reorder buffer を採用した．reorder buffer によりレジスタなどの更新を in-order にうことで，正確なマシン状態を保証する．

(d) 選択的命令無効化方式の提案：分岐予測が外れた際に行うパイプライン復元処理は，従来パイプライン・フラッシュによってパイプライン内にフェッチされているすべての命令を無効にしていた．スーパースカラではパイプライン・ステージが多く，分岐予測が外れた場合でも分岐先が既にパイプライン内にフェッチされている可能性がある．そこで，真に無効にすべき命令のみを無効にする選択的命令無効化を考案し提案した．

シミュレーションによる性能評価の結果，以下の事実が判明した．

(a) パイプライン・スライス・プロセッサ構成方式の問題点：本方式を実現するには，パイプライン間のバス，ハードウェア資源のスライス法など考慮すべき点は残されている．特に，スーパースカラでは命令間の依存関係により，パイプライン間の結合が密であるため，実現には問題が残る．

(b) reorder buffer の問題：正確なマシン状態を確保するために導入した reorder buffer が，動的コード・スケジューリングの効果を阻害している事実が判明した．特に，命令ブロック単位の reorder buffer の解放方法により，実行の遅い命令の存在を冗長させパイプライン・インターロックを頻発させる原因となった．また，waiting buffer で必要な処理にかかる時間がパイプライン・ステージとして必要であった．さらに，動的コード・スケジューリングを行うために大量のハードウェア・コストが必要であった．

(c) 動的コード・スケジューリング・アルゴリズムにおける問題：動的コード・スケジューリング・アルゴリズムのうち，演算終了後のコミット条件が，フロー依存関係にある先行命令から後続命令へのデータ・バイパスを阻害し問題となった．さらに，動的

コード・スケジューリングのみでは，命令レベルの並列性は 2〜4 程度しか得られないことが判明した．

(d) 多重依存関係表現法 ： この方法を用いた動的コード・スケジューリング・アルゴリズムには，プロセッサのハードウェア的な問題点があったために有効には使用できなかった．しかし，この多重依存関係表現法は汎用的に利用できる方式であるといえる．

(e) 選択的命令無効化方式の評価：ハードウェア的にも時間的にもコストがかかり，また，これが実行される確率はわずかでしかなく，スーパースカラ度が 4 程度では効果が現れないことが判明した．しかしながら，スーパースカラ度がさらに大きくなる場合，その効果への期待は残る．

(f) 多重命令供給：命令フェッチにおける並べ替えの問題は “可変バウンダリ・フェッチかつラインクロス有り” の構成ではハードウェア・コストが大きいにもかかわらず，“可変バウンダリ・フェッチかつラインクロスなし” と性能が変わらないことをが明らかになった．

以上の結果から，DD 型スーパースカラの開発に関して改良の余地が残された．

(3) DS 型スーパースカラ・プロセッサの設計および性能評価

DDUS プロセッサ開発の結果を踏まえて，ハードウェア的に軽量化を図り，DS 型スーパースカラ方式に基づく試作プロセッサである “DSNS プロセッサ” の設計および性能評価を行った．設計では特に，DDUS プロセッサにおけるボトルネックの原因をすべて取り払うようにしている．

(a) 静的コード・スケジューリングの積極的な利用：動的コード・スケジューリングのうち，静的コード・スケジューリングで行えることは静的に行う．これにより，ハードウェアの軽量化を図った．ただし，オブジェクト・コードの互換性を確保するため，動的ハザード解消は行う．また，アルゴルズムとしては Thornton スコアボードを用い，データのバイパスを行うことでフロー依存に伴う遅延を最小にを抑えた．

(b) 静的分岐予測と分岐先バッファを組み合わせた命令供給方式の提案：分岐頻度に偏りがある場合，静的に分岐予測が可能である．また，分岐先バッファと組み合わせることで，分岐予測がヒットしている場合遅延なしで命令の供給を行うことができる．

さらに，分岐予測が外れた場合でも，分岐遅延は最大2サイクルに抑えることが可能である．

(c) 先行条件決定方式の1実現方式の提案：先行条件決定方式の具体的導入方法を考察し提案した．先行条件決定方式は，分岐命令のコストを低く抑えると共に，静的コード・スケジューリングに対して大きな自由度を与える特長がある．また，DSNSプロセッサにおいて，演算命令のコンディションを各々のレジスタに付加したタグを用い，従来のコンディション・コード・レジスタに見られるアクセス競合を回避する方法を提案した．

(d) weakly precise interrupt方式の1実現方式の提案：DDUSでは，正確な分岐機構を保証するために，reorder buffeを採用していたが，reorder bufferがボトルネックの直接原因であった．これに代わる方式としてweakly precise interruptに着目し，1実現方式としてIPRS(ImPrecise, but ReStartable)割り込み方式を提案した．

(e) futuer fileの1実現方式の提案：future fileの1実現方式として，2重化レジスタ・ファイルを提案した．この方式により，投機的実行を行う場合でも，分岐予測が外れた際のパイプライン復元処理を可能にする．つまり，正確な分岐機構を導入できる．また，2重化レジスタ・ファイルを積極的に利用でき，静的コード・スケジューリングの自由度を大きくできるブースティングを導入した．

(f) 依存解析情報を含むロード/ストア命令の提案：ロード/ストア命令に静的スケジュールの情報を含ませる方法を提案した．

性能評価の結果以下のことが判明した．

(a) 最適化なし，および，局所コード・スケジューリングのみを施したベンチマークを使用した場合，スーパースカラ度を上げても性能は約1.5倍程度しか向上しない．しかし，ループ最適化を施すことにより平均して3倍，最も並列度の抽出できたベンチマークに対しては，約5.8倍の性能が得られた．

(b) 今回のシミュレーションでは，DSNSに採用した機能，特にブースティングおよびfutureを十分に使い切っていない．これは，静的最適化の難しさに起因している．スーパースカラ用の高度な最適化コンパイラの必要性が改めて認識させられた．

(c) DSNSプロセッサはスーパースカラ度が4の場合に性能が発揮できるようにバランスを考慮した演算器構成を採用しており，静的コード・スケジューリングを行うことで，最大限に性能を発揮できる，しかしながら，あるベンチマークによってはスケジューリングを行うことにより演算器構成によって性能が抑えられてしまう場合も発生した．このことは，処理するアプリケーションを十分に処理できるように，機能ユニットなどのハードウェア資源を最適に設計しなければならないことを意味している．

9.2 今後の課題

以下に，今後の課題について述べる．

DSNSプロセッサの性能評価を行い，十分な性能が得られることは判明した．しかしながら以下の点について，まだ議論の余地がある．

(a) 多様な応用分野での評価：今回はシミュレーションに採用したベンチマークプログラムの性格上，“数値計算処理向きプロセッサ”に限定した話しといえる．今後スーパースカラの応用分野の拡大を図るためにも，様々なアプリケーションに関してプロセッサの評価を行わなければならない．

(b) DSNSプロセッサの機能を十分に利用した評価：今回の評価ではDSNSに採用したboost命令および2重化レジスタ・ファイルを十分に生かしていない．特に2重化レジスタ・ファイルはブースティングが十分に活用されてこそ，その存在が発揮されるものだからである．

(c) 最適化コンパイラの開発：スーパースカラ・プロセッサの性能を最大に発揮するためには最適化コンパイラが不可欠である．特に，広域コード・スケジューリングである，広域コード移動およびループ再構成に関する技術を十分に採り入れていく必要がある．また，性能評価を行うに際に必要な最適化されたプログラムを得るためにも強力な最適化コンパイラが早急に望まれる．

(d) VLSI化：プロセッサの開発に際しては，VLSI化は最終目標の一つでもある．しかしながら，手軽にVLSI化が行えるような開発環境がないため，VLSI化に関して十分に議論を行っていない．今後VLSI化について十分に考慮する必要がある．将来VLSI化においてさらに多くのトランジスタを使える状況においては，DSU型スーパースカラ構成のプロセッサも採用され得ると考えられる．

謝辞

本研究の機会を与えて頂いた富田眞治教授に心から感謝致します．論文をまとめるに当たって御指導頂いた安浦寛人教授，雨宮真人教授，有川節夫教授，牛島和夫教授に感謝致します．研究の初期段階で数々の助言を戴いた末吉敏則助教授に感謝致します．数々の討議の中で熱心に御討議下さった福田 晃助教授ならびに村上和彰助手に感謝致します．特に，本研究を直接御指導して戴いた村上助手には，筆者の考えを適切な助言によって方向付けて戴き大変感謝致します．

また，本スーパースカラ・プロジェクトにおいて初期設計に携わった五島龍宏氏，筆者と共に試作プロセッサおよび最適化コンパイラの設計・開発・評価に従事して頂いた入江直彦氏，弘中哲夫氏，音成 幹氏，原 哲也氏，納富 昭氏に感謝します．

最後に，日頃から討議に参加し御討論頂く研究室の皆さんに感謝致します．

本研究は，上記の方々の御支援なしには，成し遂げられなかったことを記し，ここに謝意を表します．

参考文献

アーキテクチャ関連文献

[**Ditzel87**] Ditzel, D.R. and McLellan, H.R. : "Branch Folding in the CRISP Microprocessor: Reducing Branch Delay to Zero," *Proc. 14th Ann. Int'l. Symp. Computer Architecture*, pp.2–9, June 1987.

[**Fisher83**] Fisher, J. A. : "Very Long Instruction Word Architectures and ELI-512," *Proc. 10th Int'l. Symp. Comput. Archit.*, pp.140–150, 1983.

[**Flynn66**] Flynn, M.J. : "Very High-Speed Computing Systems," *Proc. IEEE*, vol.54, no.12, pp.1901–1909, Dec. 1966.

[**HePa90**] Hennessy, J.L. and Patterson, D.A., *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[**Hennessy89**] Hennessy, J. : "Beyond RISC," *Unix Review*, vol.7, no.9, pp.48–54, 1989.

[**HwangBriggs87**] Hwang, K. and Briggs, F.A. : *Computer Architecture and Parallel Processing*, McGraw-Hill Book Company, 1985.

[**Hwu87**] Hwu, W.W. and Patt, Y.N. : "Checkpoint Repair for High-Performance Out-of-Order Execution Machines," *IEEE Trans. Comput.*, vol.C-36, no.12, pp.1496–1514, Dec. 1987.

[**Johnson89**] Johnson, W.M. : "Super-Scalar Processor Design," *Computer Systems Laboratory Technical Report No. CSL-TR-89-383*, Stanford University, June 1989.

[**Johnson90**] Johnson, M. : *Superscalar Processor Design*, Prentice Hall, 1991.

[**Jouppi89a**] Jouppi, N.P. : "Available Instruction-Level Paralleliism for Superscalar and Superpipelined Machines," *Proc. ASPLOS-III*, pp.272–282, Apr. 1989.

[**Jouppi89b**] Jouppi, N.P. : "The Nonuniform Distribution of Instruction-Level and Machine Paralleliism and Its Effect on Performance," *IEEE Trans. Comput.*, vol.38, no.12, pp.1645–1658, Dec. 1989.

[**Kroft81**] Kroft, D. :"Lockup-free Instruction Fetch/Prefetch Cache Organization," *Proc. 8th Int'l. Symp. Comput. Archit.*, pp.81–87, May. 1981.

[**Lee84**] Lee, J.K.F. and Smith, A.J. : "Branch Prediction Strategies and Branch Target Buffer Design," *Computer*, vol.17, no.1, pp.6–22, Jan. 1984.

[**Lilja88**] Lilja, D.J. : "Reducing the Branch Penalty in Pipelined Processors, " *Computer*, vol.21, no.7, pp.47–55, July 1988.

[**Matick89**] Matick, R.E. : "Functional Cache Chip for Improved System Performance," *IBM Journal of Research and Development*, vol.33, no.1, pp.15–32, Jun. 1989.

[**McFarling86**] McFarling, S. and Hennessy, J. : "Reducing the Cost of Branches," *Proc. 13th Int'l. Symp. Computer Architecture*, pp.396–403, June 1986. y

[**Patterson**] Patterson, D.A et. al. : "A VLSI RISC," *IEEE Compuer*, pp.8–21, Sep. 1982.

[**Pleszkun87**] Pleszkun, A.R. et al. : "WISQ: A Restartable Architecture Using Queues," *Proc. 14th Ann. Int'l. Symp. Computer Architecture*, pp.290–299, June 1987.

[**Rosocha79**] Rosocha, W.G. and Lee, E.S. : "Performance Enhancement of SISD Processors," *Proc. 6th Ann. Symp. Computer Architecture*, pp.216–231, Apr. 1979.

[**SmithJ81**] Smith, J.E. : "A Study of Branch Prediction Strategies," *Proc. 8th Ann. Symp. Computer Architecture*, pp.135–148, May 1981.

[**SmithJ87**] Smith, A.J. : "Line(Block) Size Choice for CPU Cache Memories," *IEEE Trans. Comput.*, vol.C-36, no.9, Sep. 1987.

[**SmithJ88**] Smith, J.E. and Pleszkun, A. R. : "Implementing Precise Interrupts in Pipelined Processors," *IEEE Trans. Comput.*, vol.37, no.5, pp.562–573, May 1988.

[**SmithJ89**] Smith, J.E. : "Dynamic Instruction Scheduling and the Astronautics ZS–1," *Computer*, vol.22, no.7, pp.21–35, July 1989.

[**SmithM89**] Smith, M.D., Johnson, M., and Horowitz, M.A. : "Limits on Multiple Instruction Issue," *Proc. ASPLOS–III*, pp.290–302, Apr. 1989.

[**SmithM90**] Smith, M.D., Lam, M.S., and Horowitz, M.A. : "Boosting Beyond Static Scheduling in a Superscalar Processor," *Proc. 17th Int'l. Symp. Computer Architecture*, pp.344–354, May 1990.

[**Sohi87**] Sohi, G.S. and Vajapeyam, S. : "Instruction Issue Logic for High-Performance, Interruptable Pipelined Processors," *Proc. 14th Int'l. Symp. Computer Architecture*, pp.27–34, June 1987.

[**Sohi90**] Sohi, G. and Franklin, M. :"High-Bandwidth Data Memory System for Superscalar Processors," *Computer Sciences Department University of Wisconsin-Madison, Computer Sciences Technical Report No.968*, Sep. 1990.

[**Thornton64**] Thornton, J.E. : "Parallel Operation in the Control Data 6600," *Proc. Fall Joint Computer Conf.*, vol.26, pp.33–40, 1964.

[**Tomasulo67**] Tomasulo, R.M. : "An Efficient Algorithm for Exploiting Multiple Arithmetic Units," *IBM J. Res. Dev.*, vol.11, pp.25–33, Jan. 1967.

[**Weiss84**] Weiss, S. and Smith, J.E. : "Instruction Issue Logic in Pipelined Supercomputers," *IEEE Trans. Comput.*, vol.C-33, no.11, pp.1013–1022, Nov. 1984.

コンパイラ関連文献

[**Aiken88**] Aiken, A. and Nicolau, A. : "A Development Environment for Horizontal Microcode," *IEEE Trans. Soft. Eng.*, vol.14, no.5, pp.584–594, May 1988.

[**ChowF84**] Chow, F. and Hennesy, J, : "Register Allocation by Priority-Based Coloring, " *Proc. SIGPLAN'84 Symp. Compiler Construction*, June 1984.

[**ChowP89**] Chow, P. : *The MIPS-X RISC Microprocessor*, Kluwer Academic Publishers, 1989.

[**Coffman76**] Coffman, E.G., Jr. (Ed.) : *Computer and Job-Shop Scheduling Theory*, John Wiley & sons, 1976.

[**Colwell88**] Colwell, R. P. et al."A VLIW Architecture for a Trace Scheduling Compiler, " *IEEE Trans. Comput.*, vol.37, no.8, pp.967–979, Aug. 1988.

[**Ellis86**] Ellis, J.R. : *Bulldog : A Compiler for VLIW Architectures*, MIT Press 1986.

[**Fisher81**] Fisher, J. A. : "Trace Scheduling: A Technique for Global Microcode Compaction," *IEEE Trans. Comput.*, vol.C-30, no.7, pp.478–490, July 1981.

[**Golumbic90**] Golumbic, M.C. and Rainish, V. "Instruction Scheduling beyond Basic Blocks," *IBM J. Res. Dev.*, vol.34, no.1, pp.93–97, Jan. 1990.

[**Gross82**] Gross, T.R. and Hennessy, J.L. : "Optimizing Delayed Branches, " *Proc. 15th Annu. Workshop on Microprogram*, pp.114–120, Oct. 1982.

[**Hennessy83**] Hennessy, J.L. and Gross, T.R. : "Postpass Code Optimization of Pipelined Constraints," *ACM Trans. Programming Languages and Syst.*, vol.5, no.3, pp.422–448, July 1983.

[**Krishnamurthy90**] Krishnamurthy, S. : "A Brief Survey of Papers on Scheduling for Pipelined Processors," *SIGPLAN NOTICES*, vol.25, no.7, pp.97-106, July 1990.

[**久野 90**] 久野 靖 : "新しいアーキテクチャとコンパイラ技術," 情報処理, vol.31, no.6, pp.727–735, 1990 年 6 月.

[**Lam88**] Lam, M. : "Software Pipelining: An Effective Scheduling Technique for VLIW Machines," *Proc. ACM SIGPLAN'88 Conf. Programming Language Design and Implementation*, pp.318-328, June 1988.

[**中谷 90**] 中谷登志男 : "VLIW 計算機のためのコンパイラ技術," 情報処理, vol.31, no.6, pp.763–772, 1990 年 6 月.

[**村上 91**] 村上和彰 : "スーパースカラ・プロセッサの性能を最大限に引き出すコンパイラ技術," 日経エレクトロニクス, no.487, pp.165–185, 1991 年 3 月.

[**Touzeau84**] Touzeau, R. F. : "A Fortran Compiler for the FPS-164 Scientific Computer, " *Proc. ACM SIGPLAN'84 Symp. Compiler Construction*, pp.48–57, June 1984.

[**Warren90**] Warren, H.S., Jr. "Instruction Scheduling for the IBM RISC System/6000 Processor," *IBM J. Res. Dev.*, vol.34, no.1, pp.85–92, Jan. 1990.

マニュアル類

[**AMD87**] Advanced Micro Devices, *Am29000 Streamlined Instruction Processor User's Manual*, Sunnyvale, CA, 1987.

[**Fujitsu87**] Fujitsu, *MB86900 RISC Processor Architecture Manual*, FUJITSU Microelectronics Inc., Nov. 1987.

[**IBM90**] IBM Corp., *IBM RISC System/6000 Technology*, Austin, TX, 1990.

[**Intel89**] Intel Corp., *80960CA User's Manual*, Santa Clara, CA, 1989.

スーパースカラ・プロジェクト関連

DDUS 関連

[**五島 88**] 五島龍宏 : "『新風』 : SIMP 方式に基づくプロトタイプ・プロセッサ," 九州大学大学院総合理工学研究科修士論文, 1988 年 2 月

[**原 89**] 原哲也 : "『新風』プロセッサの命令供給機構に関する設計," 九州大学工学部卒業論文, 1989 年 2 月

[**原 90a**] 原ほか : "SIMP(単一命令流/多重命令パイプライン) 方式に基づくスーパースカラ・プロセッサ『新風』の命令供給機構," 情報処理学会研究会報告, 90-ARC-80-7 1990 年 1 月.

[**入江 88**] 入江ほか : "SIMP(単一命令流/多重命令パイプライン) 方式に基づく『新風』プロセッサの高速化技法および性能予測," 情報処理学会研究会報告, 88-ARC-73-11 1988 年 11 月.

[**久我 89a**] 久我ほか : "SIMP(単一命令流/多重命令パイプライン) 方式に基づく『新風』プロセッサの低レベル並列処理アルゴリズム," 並列処理シンポジウム JSPP'89 論文集, pp.163–170, 1989 年 2 月; 情報処理学会論文誌, vol.30, no.12, pp.1603–1611, 1989 年 12 月.

[**久我 89b**] 久我守弘 : "SIMP(単一命令流/多重命令パイプライン) 方式に基づく試作プロセッサ『新風』の設計," 九州大学大学院総合理工学研究科修士論文, 1989 年 2 月.

[**久我 90**] 久我ほか : "SIMP(単一命令流/多重命令パイプライン) 方式に基づくスーパースカラ・プロセッサ『新風』の性能評価," 並列処理シンポジウム JSPP'90 論文集, pp.337–344, 1990 年 5 月; 情報処理学会論文誌, vol.32, no.7, pp., 1991 年 7 月.

[**納富 90**] 納富 昭 : "スーパースカラ・プロセッサ『新風』のデータ供給機構," 九州大学工学部卒業論文, 1990 年 2 月.

[村上 88] 村上ほか:"SIMP(単一命令流/多重命令パイプライン) 方式の構想," 情報処理学会研究会報告, 88-CA-69-4, 1988 年 1 月.

[**Murakami89**] Murakami, K., Irie, N., Kuga, M., and Tomita, S. : "SIMP(Single Instruction stream/Multiple instruction Pipelining): A Novel High-Speed Single-Processor Architecture," *Proc. 16th Int'l. Symp. Computer Architecture*, pp.78-85, May 1989.

DSNS 関連

[**Kuga91**] Kuga, M., Murakami, K. and Tomita, S : "DSNS(Dynamically-hazard-resollved, Statically-code-scheduled, Nonuniform Superscalar) : Yet Another Superscalar Processor Architecture," *ACM SIGARCH Computer Architecture News*, Vol.19, No.4, pp.14-29, June 1991.

[**原 90b**] 原ほか : "SIMP(単一命令流/多重命令パイプライン) 方式に基づく改良版スーパースカラ・プロセッサの構成と処理," 電子情報通信学会技術研究報告, CPSY-90-55 1990 年 7 月.

[**原 91a**] 原ほか :"DSN 型スーパースカラ・プロセッサ・プロトタイプの分岐パイプライン," 情報処理学会研究会報告, ARC-86-3, 1991 年 1 月.

[**原 91b**] 原 哲也 :"DSN 型スーパースカラ・プロセッサ・プロトタイプの設計," 九州大学大学院総合理工学研究科修士論文, 1991 年 2 月.

[**村上 90**] 村上ほか : "SIMP(単一命令流/多重命令パイプライン) 方方式に基づくスーパースカラ・プロセッサの改良方針," 電子情報通信学会技術研究報告, CPSY-90-54, 1990 年 7 月.

[**納富 91a**] 納富ほか : "DSN 型スーパースカラ・プロセッサ・プロトタイプのロード/ストア・パイプライン," 情報処理学会研究会報告, ARC-86-4, 1991 年 1 月.

[**納富 91b**] 納富ほか : "DSN 型スーパースカラ・プロセッサ・プロトタイプ —アーキテクチャおよび静的コード・スケジューリングに関する総合評価—," 情報処理学会並列処理シンポジウム'91, 1991 年 5 月.

付録 A

種々のスーパースカラの仕様

(a) **研究用試作機 (大学)**

- Kyushu DDUS
- Kyushu DSNS
- Stanford TORCH
- Stanford MATCH

(b) **研究用試作機 (企業)**

- Mitsubishi OMEGA
- Mitsubishi SARCH
- Hitachi On-chip Multiple Superscalar Processor
- Metaflow Lightning

(c) **商用機**

- IBM RISC System/6000
- Intel i960CA
- National Semiconductor Swordfish
- Motorola 88110

Table A.1　研究用試作機 (大学その 1)

		Kyushu DDUS	Kyushu DSNS
Category		DDU	DSN
Instruction Fetch	Degree of Fetch	4	4
	Fetch Boundary	Variable	Fixed
	Line Crossing	Disallowed	–
	Line Size	16 Instructions	16 Instructions
Functional Units	Uniformity	Uniform	Nonuniform
	Type×Number	a. General-purpose×4 ・Integer ALU (include Load/Store, Branch ・Integer Multiplier ・Floating-point ALU ・Floating-point Mul.	a. Integer×4 b. Floating-point×4 c. Load/Store×2 d. Branch×1 e. Control×2
Instruction Issue	Degree of issue	4	4
	Class×Number	a. Any×4	a. Integer×2 b. Floating-point×2 c. Load/Store×2 d. Branch×1 e. Control×2
When Hazard Resolved?		Dynamic	Dynamic
Code Scheduling	When?	Dynamic	Static
	Techniques	Extended Tomasulo Algorithm	–
Ordering	Execution Order	Out-of-order	In-order
	Completion Order	Out-of-order	Out-of-order
Branch Handling	Policy	Dynamic Branch Prediction + BTB	Static Branch Prediction + BTB
	Speculative Execution	Conditional Mode + Reorder Buffer	Conditional Mode + Dual Register File
	Recovery	Selective Squashing	Pipeline Flush
Precise-interrupt Scheme		Reorder Buffer	Weakly Precise Interrupt
First Announce		1987	1990

Table A.2　研究用試作機 (大学その 2)

		Stanford TORCH	Stanford MATCH
Category		SSN	DDN
Instruction Fetch	Degree of Fetch	2	2
	Fetch Boundary	Fixed	Fixed
	Line Crossing	–	–
	Line Size	?	?
Functional Units	Uniformity	Nonuniform	Nonuniform
	Type×Number	a. Integer×3 b. Floating-point×4 c. Load/Store×2 d. Branch×1	a. Integer×3 b. Floating-point×4 c. Load×2 d. Store×2 e. Branch×1
Instruction Issue	Degree of issue	2	2
	Class×Number	a. Integer×2 b. Floating-point×2 c. Load/Store×2 d. Branch×1	a. Integer×2 b. Floating-point×2 c. Load×2 d. Store×2 e. Branch×1
When Hazard Resolved?		Static	Dynamic
Code Scheduling	When?	Static	Dynamic
	Techniques	–	Tomasulo Algorithm
Ordering	Execution Order	In-order	Out-of-order
	Completion Order	Out-of-order	Out-of-order
Branch Handling	Policy	Static Branch Prediction + Delayed Branch	Dynamic Branch Prediction + BTB
	Speculative Execution	Boosting + Shadow Structure	Conditional Mode + Reorder Buffer
	Recovery	Pipeline Flush	Pipeline Flush
Precise-interrupt Scheme		Future File	Reorder Buffer
First Announce		1990	1990

Table A.3　研究用試作機 (企業その 1)

		Matsushita OMEGA	Mitsubishi SARCH
Category		DSN	DSN
Instruction Fetch	Degree of Fetch	2	4
	Fetch Boundary	?	Fixed
	Line Crossing	?	–
	Line Size	4 Instructions	?
Functional Units	Uniformity	Nonuniform	Nonuniform
	Type×Number	a. Integer, Load/Store×1 b. Integer, Store×1 c. Floating-point×2 d. Branch×1	a. Integer×3 b. Load/Store×1 c. Branch×1
Instruction Issue	Degree of issue	2	4
	Class×Number	a. Integer×2 b. Load×1 c. Store×2 d. Floating-point×2 e. Branch×1	a. Integer×3 b. Load/Store×1 c. Branch×1
When Hazard Resolved?		Dynamic	Dynamic
Code Scheduling	When?	Static	Static
	Techniques	–	–
Ordering	Execution Order	Out-of-order	In-order
	Completion Order	Out-of-order	Out-of-order
Branch Handling	Policy	Static Branch Prediction + Delayed Branch	Static Branch Prediction + Delayed Branch
	Speculative Execution	–	Boosting
	Recovery	–	Pipeline Flush
Precise-interrupt Scheme		Directly Tag Compare	?
First Announce		1991	1991

Table A.4　研究用試作機 (企業その 2)

		Hitachi On-chip Multiple Superscalar Proccesors†	Metaflow Lightning
Category		DSN	DDN
Instruction Fetch	Degree of Fetch	2	4
	Fetch Boundary	?	Variable
	Line Crossing	?	Allowed
	Line Size	?	8 Instructions
Functional Units	Uniformity	Nonuniform	Nonuniform
	Type×Number	a. Integer, Load/Store×1 b. Integer, Branch×1	a. Integer×2 b. Floating-point×2 c. Load/Store×1 d. Branch×1
Instruction Issue	Degree of issue	2	?
	Class×Number	a. Integer×2 b. Floating-point×2 c. Load/Store×2 d. Branch×1	a. Integer×2 b. Floating-point×2 c. Load/Store×1 d. Control×2
When Hazard Resolved?		Dynamic	Dynamic
Code Scheduling	When?	Static	Dynamic
	Techniques	–	Register Renaming
Ordering	Execution Order	In-order	Out-of-order
	Completion Order	In-order	Out-of-order
Branch Handling	Policy	–	Dynamic Branch Prediction + BTB(?)
	Speculative Execution	–	Conditional Mode + Reorder Buffer
	Recovery	–	Pipeline Flush
Precise-interrupt Scheme		–	Reorder Buffer
First Announce		1991	1991

†A chip contains two superscalar processors.

Table A.5　商用機 (その 1)

		IBM RISC System/6000	Intel i960CA
Category		DDN	DSN
Instruction Fetch	Degree of Fetch	4	3 or 4
	Fetch Boundary	Variable	Variable
	Line Crossing	Allowed	Allowed
	Line Size	16 Instructions	8 Instructions
Functional Units	Uniformity	Nonuniform	Nonuniform
	Type×Number	a. Integer, Load/Store×1 b. Floating-point×1 c. Branch×1 d. Control×1	a. Integer×1 b. Load/Store×1 c. Branch×1
Instruction Issue	Degree of issue	4	3
	Class×Number	a. Integer, Load/Store×1 b. Floating-point×1 c. Branch×1 d. Control×1	a. Integer×1 b. Load/Store×1 c. Branch×1
When Hazard Resolved?		Dynamic	Dynamic
Code Scheduling	When?	Dynamic	Static
	Techniques	Register Renaming	–
Ordering	Execution Order	In-order	In-order
	Completion Order	Out-of-order	Out-of-order
Branch Handling	Policy	Predict-untaken	Static Branch Prediction
	Speculative Execution	Conditional Mode	Conditional Mode
	Recovery	Pipeline Flush	Pipeline Flush
Precise-interrupt Scheme		Guarantee-not-interrupted	–
First Announce		1990	1990

Table A.6　商用機 (その 2)

		NS Swordfish	Motorola 88110
Category		DSN	DS(D)N†
Instruction Fetch	Degree of Fetch	1 or 2	2
	Fetch Boundary	Fixed	Variable
	Line Crossing	–	Allowed(?)
	Line Size	1 or 2 Instructions (8 bytes)	8 Instructions
Functional Units	Uniformity	Nonuniform	Nonuniform
	Type×Number	a. Integer, Load/Store×1 b. Integer, Load/Store, Floating-point, Branch×1	a. Integer×4 b. Floating-point×2 c. Graphics×2 d. Load/Store×1 e. Branch×1
Instruction Issue	Degree of issue	2	2
	Class×Number	a. Integer×2 b. Floating-point×1 c. Load/Store×2 d. Branch×1	a. Integer×2 b. Floating-point×2 c. Graphics×2 d. Load/Store×1 e. Branch×1
When Hazard Resolved?		Dynamic	Dynamic
Code Scheduling	When?	Static	Static (+ Dynamic)†
	Techniques	–	Reservation Stations†
Ordering	Execution Order	In-order	In-order
	Completion Order	In-order	Out-of-order
Branch Handling	Policy	Predict-taken	Dynamic/Static(?) Branch Prediction + BTB
	Speculative Execution	–	Conditional Mode + History Buffer
	Recovery	–	Pipeline Flush
Precise-interrupt Scheme		–	History Buffer
First Announce		1991	1991

†The order of Load/Store instructions are scheduled dynamically using store reservation stations.

付録 B

試作プロセッサの命令一覧

凡例

B	: boost bit	*	: 0 or 1
OP.code	: Operation code	+	: add
GR	: General Register	-	: subtract
FR.s	: Floating-point Register (single)	×	: multiply
FR.d	: Floating-point Register (double)	/	: divide
TR	: True/False Register	&	: logical and
TBR	: Trap Base Register	|	: logical or
PC	: Program counter	⊗	: exclusive or
Reg.	: Register	$a?b:c$	: if a then b else c
Imm.n	: n bits Immediate	¬	: complement
Cond.	: Condition	$\|n\|$	: absolute
.tag	: Tag field of Register	≪	: left shift
FU	: Functional Unit	≫	: rgiht shift
d	: don't care	W	: Wrapped Numbers
		U	: Unrounded Normalized Numbers
		D	: Denormalized Numbers

B.1 コンディション・フィールドの詳細

GR のコンディション・フィールド (GR タグ) は 4 ビット長で，以下のフィールドを持つ．

N	Z	V	C

N(negative): Sign Flag
Z(zero): Zero Flag
V(oVerflow): Overflow Flag
C(Carry): Carry Flag

FR のコンディション・フィールド (FR タグ) は 4 ビット長で，以下のフィールドを持つ．なお，保持される内容は比較命令と算術命令とでは異なり，算術命令では例外情報がセットされる．

意味

tag	比較命令	例外情報
0000	Equal	Result = +0 or −0, exact
0001	Less than	Result = +infinity or −infinity, exact
0010	Greater than	Result finite and ≠ 0, exact
0011		Result finite and ≠ 0, inexact
0100		− not used
0101		Overflow & inexact
0110		Underflow
0111		Underflow & inexact
1000		Operand A is denormalized
1001		Operand B is denormalized
1010		Operand A & B are denormalized
1011		Divide by zero
1100		Operand A is NaN†
1101		Operand B is NaN
1110		Operand A & B are NaN
1111	Unorderd	Invalid Operation

† Not a Number

B.2　分岐条件決定における条件一覧

・GR レジスタのテスト条件

Mnemonic	Condition	Code	Test Operation
T	True	0000	1
F	False	0001	0
HI	High	0010	$\overline{C \cdot Z}$
LS	Low or Same	0011	$\overline{C + Z}$
NC	Not Carry	0100	$\overline{C}$
C	Carry	0101	C
NZ	Not Zero	0110	$\overline{Z}$
Z	Zero	0111	Z
NV	Not Overflow	1000	$\overline{V}$
V	Overflow	1001	V
P	Plus	1010	N
M	Minous	1011	N
GE	Greater or Equal	1100	$N \cdot V + \overline{N} \cdot \overline{V}$
LT	Less Than	1101	$N \cdot \overline{V} + \overline{N} \cdot V$
GT	Greater Than	1110	$N \cdot V \cdot \overline{Z} + \overline{N} \cdot \overline{V} \cdot \overline{Z}$
LE	Less or Equal	1111	$N + N \cdot V + \overline{N} \cdot V$

$\cdot$: logical and　　$+$: logical or　　$\overline{X}$: not

・FR レジスタのテスト条件

Mnemonic	Condition	Code	Test Operation
T	True	0000	1
F	False	0001	0
U	Unordered	0010	U
G	Greater	0011	G
UG	Unordered or Greater	0100	$G + U$
L	Less	0101	L
UL	Unordered or less	0110	$L + U$
LG	Less or Greater	0111	$L + G$
NE	Not Equal	1000	$L + G + U$
E	Equal	1001	E
UE	Unordered or Equal	1010	$E + U$
GE	Greater or Equal	1011	$E + G$
UGE	Unordered or Greater or Equal	1100	$E + G + U$
LE	Less or Equal	1101	$E + L$
ULE	Unordered or Less or Equal	1110	$E + L + U$
O	Ordered	1111	$E + L + G$

$+$: logical or

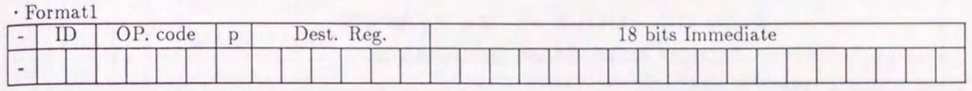
U : Unordered (1111)　　E : Equal (0000)

L : Less (0001)　　G : Greater (0010)

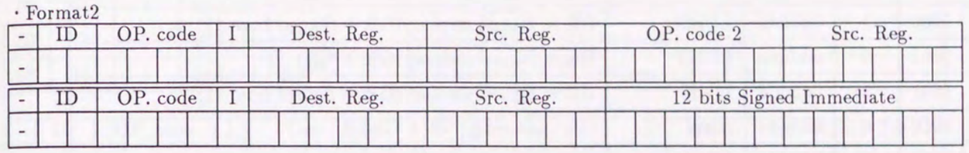
B.3　DDUS プロセッサの命令一覧

Figure B.1　DDUS プロセッサの命令フォーマット

・Format1



・Format2



・Format3

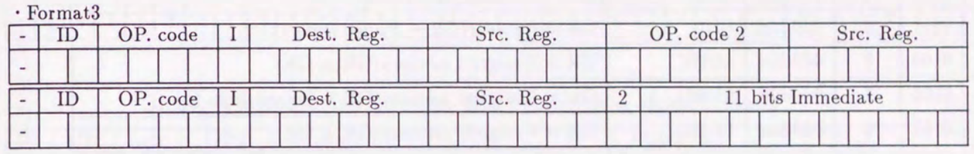


Table B.1　DDUS プロセッサの命令表

Integer Load/Store Instructions

ID	OP	Imm	OP2	Mnemonic	Operation
00	0000	0	0ddddd	LDBS	GR = Signed memory(GR + GR)
00	0000	1	0ddddd	LDBS	GR = Signed memory(GR + Signed Imm.11)
00	0001	0	0ddddd	LDHS	GR = Signed memory(GR + GR)
00	0001	1	0ddddd	LDHS	GR = Signed memory(GR + Signed Imm.11)
00	0010	0	0ddddd	LDW	GR = memory(GR + GR)
00	0010	1	0ddddd	LDW	GR = memory(GR + Signed Imm.11)
00	0011	0	0ddddd	LDL	GR = memory(GR + GR)
00	0011	1	0ddddd	LDL	GR = memory(GR + Signed Imm.11)
00	0100	0	0ddddd	LDBU	GR = Unsigned memory(GR + GR)
00	0100	1	0ddddd	LDBU	GR = Unsigned memory(GR + Signed Imm.11)
00	0101	0	0ddddd	LDHU	GR = Unsigned memory(GR + GR)
00	0101	1	0ddddd	LDHU	GR = Unsigned memory(GR + Signed Imm.11)
00	0110	0	0ddddd	LDSR	SR = Signed memory(GR + GR)
00	0110	1	0ddddd	LDSR	SR = Signed memory(GR + Signed Imm.11)
00	0111	0	0ddddd	STSR	memory(GR + GR)
00	0111	1	0ddddd	STSR	memory(GR + Signed Imm.11)
00	1000	0	0ddddd	STB	memory(GR + GR)
00	1000	1	0ddddd	STB	memory(GR + Signed Imm.11)
00	1001	0	0ddddd	STH	memory(GR + GR)
00	1001	1	0ddddd	STH	memory(GR + Signed Imm.11)
00	1010	0	0ddddd	STW	memory(GR + GR)
00	1010	1	0ddddd	STW	memory(GR + Signed Imm.11)
00	1011	0	0ddddd	STL	memory(GR + GR)
00	1011	1	0ddddd	STL	memory(GR + Signed Imm.11)

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						OP. code 2						Src. Reg.					
-	0	0	*	*	*	*	0	*	*	*	*	*	*	*	*	*	*	*	*	0	d	d	d	d	d	*	*	*	*	*	*

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						2	11 bits Signed Immediate										
-	0	0	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*	0	*	*	*	*	*	*	*	*	*	*	*	*

Floating-point Load/Store Instructions

ID	OP	Imm	OP2	Mnemonic	Operation
00	1100	0	0ddddd	LDFS	FR.s = memory(GR + GR)
00	1100	1	0ddddd	LDFS	FR.s = memory(GR + Signed Imm.11)
00	1101	0	0ddddd	LDFD	FR.d = memory(GR + GR)
00	1101	1	0ddddd	LDFD	FR.d = memory(GR + Signed Imm.11)
00	1110	0	0ddddd	STFS	memory(GR + GR) = FR.s
00	1110	1	0ddddd	STFS	memory(GR + Signed Imm.11) = FR.s
00	1111	0	0ddddd	STFD	memory(GR + GR) = FR.d
00	1111	1	0ddddd	STFD	memory(GR + Signed Imm.11) = FR.d

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						OP. code 2						Src. Reg.					
-	0	0	1	1	*	*	0	*	*	*	*	*	*	*	*	*	*	*	*	0	d	d	d	d	d	*	*	*	*	*	*

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						2	11 bits Signed Immediate										
-	0	0	1	1	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*	0	*	*	*	*	*	*	*	*	*	*	*	*

Shift, Monadic Operation, Bit Operation Instructions

ID	OP	Imm	OP2	Mnemonic	Operation
00	0000	0	1ddddd	SHA	GR = GR ≪ GR or GR = GR ≫ GR
00	0000	1	1ddddd	SHA	GR = GR ≪ Imm.5 or GR = GR ≫ Imm.5
00	0001	0	1ddddd	SHL	GR = GR ≪ GR or GR = GR ≫ GR
00	0001	1	1ddddd	SHL	GR = GR ≪ Imm.5 or GR = GR ≫ Imm.5
00	0010	0	1ddddd	ROT	GR = GR rotate GR
00	0010	1	1ddddd	ROT	GR = GR rotate Signed Imm.6
00	0011	0	1ddddd	NEG	GR = (−GR)
00	0011	1	1ddddd		
00	0100	0	1ddddd	EXTB	GR = (GR & (1 ≪ GR)) ? 1 : 0
00	0100	1	1ddddd	EXTB	GR = (GR & (1 ≪ Imm.5)) ? 1 : 0
00	0101	0	1ddddd	SETB	GR = GR | (1 ≪ GR)
00	0101	1	1ddddd	SETB	GR = GR | (1 ≪ Imm.5)
00	0110	0	1ddddd	RESB	GR = GR & ¬(1 ≪ GR)
00	0110	1	1ddddd	RESB	GR = GR & ¬(1 ≪ Imm.5)
00	0111	0	1ddddd	PRIOR	GR = priority(GR)
00	0111	1	1ddddd		

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						OP. code 2						Src. Reg.					
-	0	0	0	*	*	*	0	*	*	*	*	*	*	*	*	*	*	*	*	1	d	d	d	d	d	*	*	*	*	*	*

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						2	11 bits Signed Immediate										
-	0	0	0	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*

Rotate, Move Operation Instructions

ID	OP	Imm	OP2	Mnemonic	Operation
00	1000	0	1ddddd	SLBZ	GR = GR $\ll$ 1
00	1000	1	1ddddd		
00	1001	0	1ddddd	SLBC	GR = GR $\ll$ 1
00	1001	1	1ddddd		
00	1010	0	1ddddd	SRBZ	GR = GR $\gg$ 1
00	1010	1	1ddddd		
00	1011	0	1ddddd	SRBC	GR = GR $\gg$ 1
00	1011	1	1ddddd		
00	1100	0	1ddddd	SRBS	GR = GR $\gg$ 1
00	1100	1	1ddddd		
00	1101	*	1ddddd		
00	1110	0	1ddddd	MOV	GR = GR
00	1110	1	1ddddd		
00	1111	0	1ddddd	MOVS	SR = SR
00	1111	1	1ddddd		

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						OP. code 2						Src. Reg.					
-	0	0	1	*	*	*	0	*	*	*	*	*	*	*	*	*	*	*	*	1	d	d	d	d	d	*	*	*	*	*	*

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						2	11 bits Signed Immediate										
-	0	0	1	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*

Dyadic Operation Instructions

ID	OP	Imm	OP2	Mnemonic	Operation
01	0000	0	0ddddd	AND	GR = GR & GR
01	0000	1	0ddddd	AND	GR = GR & Signed Imm.11
01	0001	0	0ddddd	XNOR	GR = $\neg$(GR $\otimes$ GR)
01	0001	1	0ddddd	XNOR	GR = $\neg$(GR $\otimes$ Signed Imm.11)
01	0010	0	0ddddd	ADD	GR = GR + GR
01	0010	1	0ddddd	ADD	GR = GR + Signed Imm.11
01	0011	0	0ddddd	ADDC	GR = GR + GR + Cy
01	0011	1	0ddddd	ADDC	GR = GR + Signed Imm.11 + Cy
01	0100	0	0ddddd	SUB	GR = GR $-$ GR
01	0100	1	0ddddd	SUB	GR = GR $-$ Signed Imm.11
01	0101	0	0ddddd	SUBC	GR = GR $-$ GR $-$ Cy
01	0101	1	0ddddd	SUBC	GR = GR $-$ Signed Imm.11 $-$Cy
01	0110	0	0ddddd	OR	GR = GR | GR
01	0110	1	0ddddd	OR	GR = GR | Signed Imm.11
01	0111	0	0ddddd	XOR	GR = GR $\otimes$ GR
01	0111	1	0ddddd	XOR	GR = GR $\otimes$ Signed Imm.11
01	1000	0	0ddddd	MULWS	GR = GR $\times$ GR
01	1000	1	0ddddd	MULWS	GR = GR $\times$ Signed Imm.11
01	1001	0	0ddddd	MULLS	GR = GR $\times$ GR
01	1001	1	0ddddd	MULLS	GR = GR $\times$ Signed Imm.11
01	1010	0	0ddddd	MULWU	GR = GR $\times$ GR
01	1010	1	0ddddd	MULWU	GR = GR $\times$ Unsigned Imm.11
01	1011	0	0ddddd	MULLU	GR = GR $\times$ GR
01	1011	1	0ddddd	MULLU	GR = GR $\times$ Unsigned Imm.11

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						OP. code 2						Src. Reg.					
-	0	1	*	*	*	*	0	*	*	*	*	*	*	*	*	*	*	*	*	0	d	d	d	d	d	*	*	*	*	*	*

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						2	11 bits Signed Immediate										
-	0	1	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*	0	*	*	*	*	*	*	*	*	*	*	*	*

Field Logical Operation Instructions

ID	OP	Imm	OP2	Mnemonic	Operation
01	0000	1	1ddddd	FLPAA	Yi=Ai, unselect B
01	0001	1	1ddddd	FLPBA	Yi=Bi, unselect B
01	0010	1	1ddddd	FLPAU	p≥0, Yi=Ai−p or p¡0, Yi−|p|=Ai
01	0011	1	1ddddd	FLNAA	Yi=$\bar{A}$i, unselect B
01	0100	1	1ddddd	FLNBA	Yi=$\bar{B}$i, unselect B
01	0101	1	1ddddd	FLNAU	p≥0, Yi=$\bar{A}$i−p or p¡0, Yi−|p|=$\bar{A}$i
01	0110	1	1ddddd	FLOAA	Yi=Ai|Bi, unselect B
01	0111	1	1ddddd	FLOAU	p≥0, Yi=$\bar{A}$i−p|Bi or p¡0, Yi−|p|=Ai|Bi−|p|
01	1000	1	1ddddd	FLXAA	Yi=Ai*otimes*Bi, unselect B
01	1001	1	1ddddd	FLXAU	p≥0, Yi=$\bar{A}$i−p*otimes*Bi or p¡0, Yi−|p|=Ai⊗Bi−|p|
01	1010	1	1ddddd	FLAAA	Yi=Ai&Bi, unselect B
01	1011	1	1ddddd	FLAAU	p≥0, Yi=$\bar{A}$i−p& Bi or p¡0, Yi−|p|=Ai& Bi−|p|
01	1100	1	1ddddd	FLEAU	p≥0, Yi=Ai−p or p¡0, Yi−|p|=Ai
01	1101	1	1ddddd	FLEBU	p≥0, Yi=Bi−p or p¡0, Yi−|p|=Bi
01	1110	1	1ddddd	FLEAB	
01	1111	1	1ddddd	FLEBA	

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						2	11 bits Unsigned Immediate										
-	0	1	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*

Floating-point Operation Instructions

ID	OP	Pre	OP2	Mnemonic	Operation
10	0000	0	0ddddd	FADDS	FR.s = FR.s + FR.s
10	0000	1	0ddddd	FADDD	FR.d = FR.d + FR.d
10	0001	0	0ddddd	FADDS	FR.s = |FR.s| + |FR.s|
10	0001	1	0ddddd	FADDD	FR.d = |FR.d| + |FR.d|
10	0010	0	0ddddd	FADDS	FR.s = |FR.s + FR.s|
10	0010	1	0ddddd	FADDD	FR.d = |FR.d + FR.d|
10	0011	0	0ddddd	FSUBS	FR.s = FR.s − FR.s
10	0011	1	0ddddd	FSUBD	FR.d = FR.d − FR.d
10	0100	0	0ddddd	FSUBS	FR.s = |FR.s| − |FR.s|
10	0100	1	0ddddd	FSUBD	FR.d = |FR.d| − |FR.d|
10	0101	0	0ddddd	FSUBS	FR.s = |FR.s − FR.s|
10	0101	1	0ddddd	FSUBD	FR.d = |FR.d − FR.d|
10	0110	0	0ddddd	FMULS	FR.s = FR.s × FR.s
10	0110	1	0ddddd	FMULD	FR.d = FR.d × FR.d
10	0111	0	0ddddd	FDIVS	FR.s = FR.s / FR.s
10	0111	1	0ddddd	FDIVD	FR.d = FR.d / FR.d

-	ID		OP. code		p	Dest. Reg.						Src. Reg.						OP. code 2						Src. Reg.					
-	1	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	d	d	d	d	d	*	*	*	*	*	*

Floating-point Move, Convert Instructions

ID	OP	Pre	OP2	Mnemonic	Operation
10	1100	0	0ddddd	FMOVS	FR.s = FR.s
10	1100	1	0ddddd	FMOVD	FR.d = FR.d
10	1101	0	0ddddd	FStoI	GR = FR.s
10	1101	1	0ddddd	FDtoI	GR = FR.d
10	1110	0	0ddddd	ItoFS	FR.s = GR
10	1110	1	0ddddd	ItoFD	FR.d = GR
10	1111	0	0ddddd	FStoFD	FR.s = FR.s
10	1111	1	0ddddd	FDtoFS	FR.d = FR.d

-	ID		OP. code		p	Dest. Reg.						Src. Reg.						OP. code 2						Src. Reg.					
-	1	0	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	d	d	d	d	d	*	*	*	*	*	*

Integer Conditioning Instructions

ID	OP	Imm	OP2	Mnemonic	Operation
10	0000	0	10tttt	C&T	TF = (GR – GR).tag test cond.
10	0000	1	10tttt	C&T	TF = (GR – Signed Imm.6).tag test cond.
10	0001	0	10tttt	TES	TF = GR.tag test cond.

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						O2		Cond.				Src. Reg.					
-	1	0	0	0	0	*	0	*	*	*	*	*	*	*	*	*	*	*	*	1	0	t	t	t	t	*	*	*	*	*	*

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						O2		Cond.				6 bits S. Imm.					
-	1	0	0	0	0	*	1	*	*	*	*	*	*	*	*	*	*	*	*	1	0	t	t	t	t	*	*	*	*	*	*

Floating-point Conditioning Instructions

ID	OP	Pre	OP2	Mnemonic	Operation
00	0100	0	10tttt	FC&TS	TF = (FR.s – FR.s).tag test cond.
00	0100	1	10tttt	FC&TD	TF = (FR.d – FR.d).tag test cond.
00	0101	0	10tttt	FC&TS	TF = (|FR.s| – |FR.s|).tag test cond.
00	0101	1	10tttt	FC&TD	TF = (|FR.d| – |FR.d|).tag test cond.
00	0110	0	10tttt	FTESS	TF = FR.s.tag test cond.
00	0110	1	10tttt	FTESD	TF = FR.d.tag test cond.

-	ID		OP. code				p	Dest. Reg.						Src. Reg.						O2		Cond.				Src. Reg.					
-	1	0	0	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	t	t	t	t	*	*	*	*	*	*

Branch, Call, Trap Instructions

ID	OP	Imm	OP2	Mnemonic	Operation
11	0000	0	dddddd	BRANCH	PC = (TF) ? PC+(GR ≪ 2) : PC+4
11	0000	1	******	BRANCH	PC = (TF) ? PC+(Signed Imm.12 ≪ 2) : PC+4
11	0001	*	dddddd		
11	0010	0	dddddd	CALL	GR = PC, PC = (TF) ? PC+(GR ≪ 2) : PC+4
11	0010	1	******	CALL	GR = PC, PC = (TF) ? PC+(Signed Imm.12 ≪ 2) : PC+4
11	0011	*	dddddd		
11	0100	0	dddddd	TRAP	PC = (TF) ? PC+(GR ≪ 2) : PC+4
11	0100	1	******	TRAP	PC = (TF) ? PC+(Signed Imm.12 ≪ 2) : PC+4
11	0101	1	******	RETT	PC = (TF) ? GR+(Signed Imm.12 ≪ 2) : PC+4
11	0110	*	dddddd		
11	0111	*	dddddd		

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						OP. code 2						Src. Reg.					
-	1	1	0	*	*	*	0	*	*	*	*	*	*	*	*	*	*	*	*	d	d	d	d	d	d	*	*	*	*	*	*

-	ID		OP. code				I	Dest. Reg.						Src. Reg.						12 bits Signed Immediate											
-	1	1	0	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Immediate Load Instructions

ID	OP	Pre	OP2	Mnemonic	Operation
11	1000	0	dddddd	LDIL	GR = Signed Imm.18
11	1000	1	dddddd	LDIH	GR = Signed Imm.18 ≪ 14
11	1001	0	dddddd	LDFIS	FR.s = itofs(Signed Imm.18)
11	1001	1	dddddd	LDFID	FR.d = itofd(Signed Imm.18)

-	ID		OP. code		p	Dest. Reg.						18 bits Signed Immediate																
-	1	1	1	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

B.4 DSNS プロセッサの命令一覧

Figure B.2 DSNS プロセッサの命令フォーマット

・Format1

B	Group No.	OP. code	23bit Immediate

・Format2

B	Group No.	OP. code	Dest. Reg.	13bit Immediate	Source 1 Reg.

B	Group No.	OP. code	Dest. Reg.	18bit Immediate

・Format3

B	Group No.	OP. code	Dest. Reg.	Source 1 Reg.	8bit Immediate	Source 2 Reg.

B	Group No.	OP. code	Dest. Reg.	Source 1 Reg.	18bit Immediate

・Format4

B	Group No.	OP. code	Dest. Reg.	Source 1 Reg.	Condition		Source 2 Reg.

B	Group No.	OP. code	Dest. Reg.	Source 1 Reg.	Condition	9bit Immediate

Table B.2 DSNS プロセッサの命令表

Group0 (Branch, Jump, Trap)

B	OP	Mnemonic	Operation	FU
0	0000	branch	PC = (TF) ? PC+(GR ≪ 2) : PC+4	branch
0	0001	branch	PC = (TF) ? PC+(Signed Imm.18 ≪ 2) : PC+4	branch
0	0010	branchn	PC = (TF) ? PC+(GR ≪ 2) : PC+4	branch
0	0011	branchn	PC = (TF) ? PC+(Signed Imm.18 ≪ 2) : PC+4	branch
0	0100			
0	0101	jump	PC = (TF) ? GR+(Signed Imm.18 ≪ 2) : PC+4	branch
0	0110			
0	0111	jumpn	PC = (TF) ? GR+(Signed Imm.18 ≪ 2) : PC+4	branch
0	1000			
0	1001			
0	1010	trap	PC = (TF) ? TBR+((GR & 0x7f) ≪ 4) : PC+4	branch
0	1011	trap	PC = (TF) ? TBR+(Unsigned Imm.7 ≪ 4) : PC+4	branch
0	1100			
0	1101			
0	1110			
0	1111	tret	PC = (TF) ? GR+(Signed Imm.18 ≪ 2) : PC+4	branch

† privileged instruction

・branch, branchn

B	Group No.	OP. code	TF Register	not used	Disp. Reg.
0	0 0 0 0	0 0 * 0	* * * * *		* * * * *

B	Group No.	OP. code	TF Register	18bit Signed Displacement
*	0 0 0 0	0 0 * 1	* * * * *	

・jump, jumpn

B	Group No.	OP. code	TF Register	Base Reg.	13bit Signed Displacement
0	0 0 0 0	0 1 * 1	* * * * *	* * * * *	

・trap

B	Group No.	OP. code	TF Register	not used	Source Reg.
0	0 0 0 0	1 0 1 0	* * * * *		* * * * *

B	Group No.	OP. code	TF Register	not used	Trap Number
0	0 0 0 0	1 0 1 1	* * * * *		* * * * * *

・tret

B	Group No.	OP. code	TF Register	Base Reg.	13bit Signed Displacement
0	0 0 0 0	1 1 1 1	* * * * *	* * * * *	

Group1 (TF logical & test)

B	OP	Mnemonic	Operation	FU
*	0000	tfand	TF = TF & TF	tf/test
*	0001	tfor	TF = TF | TF	tf/test
*	0010	tfxor	TF = TF ⊕ TF	tf/test
*	0011	tfand	TF = (GR.tag test cond.) & TF	tf/test
*	0100	tfor	TF = (GR.tag test cond.) | TF	tf/test
*	0101	tfxor	TF = (GR.tag test cond.) ⊕ TF	tf/test
*	0110	tfand	TF = (FR.tag test cond.) & TF	tf/test
*	0111	tfor	TF = (FR.tag test cond.) | TF	tf/test
*	1000	tfxor	TF = (FR.tag test cond.) ⊕ TF	tf/test
*	1001			
*	1010	fcat.s	TF = (FR.s − FR.s).tag test cond.	falu
*	1011	fcat.d	TF = (FR.d − FR.d).tag test cond.	falu
*	1100	cat	TF = (GR − GR).tag test cond.	ialu
*	1101	cat	TF = (GR − Signed Imm.9).tag test cond.	ialu
*	1110	aat	TF = (GR & GR).tag test cond.	ialu
*	1111	aat	TF = (GR & Signed Imm.9).tag test cond.	ialu

B	Group No.	OP. code	TF Register	Source Reg.	Condition	not used	Source. Reg.
*	0 0 0 1	* * * *	* * * *	* * * *	* * * *		* * * * *

B	Group No.	OP. code	TF Register	Source Reg.	Condition	9bit Signed Immediate
*	0 0 0 1	1 1 * 1	* * * *	* * * *	* * * *	

Group2(Move)

B	OP	Mnemonic	Operation	FU
*	0000	grcftogr	GR = GR.tag	conv
*	0001	grtogrcf	GR.tag = GR	conv
*	0010	mov	GR = GR	conv
*	0011			conv
*	1000	frcftogr	GR = FR.tag	conv
*	1001	grtofrcf	FR.tag = GR	conv
*	1010	fmov	FR.d = FR.d	conv
*	1011			conv

B	Group No.	OP. code	Dest. Reg.	Src. Reg.	not used
*	0 0 1 0	* * * *	* * * *	* * * *	

Group3 (Type Conversion)

B	OP	Mnemonic	Operation	FU
*	0000	itofs	FR.s = GR	conv
*	0001	itofd	FR.d = GR	conv
*	0010	fstoi	GR = FR.s	conv
*	0011	fdtoi	GR = FR.d	conv
*	0100	fstofd	FR.d = FR.s	conv
*	0101	fdtofs	FR.s = FR.d	conv
*	1000	ustodse	D32(exact) = U32	conv
*	1001	udtodde	D64(exact) = U64	conv
*	1010	ustodsie	D32(inexact) = U32	conv
*	1011	udtoddie	D64(inexact) = U64	conv
*	1100	dstows	W32 = D32	conv
*	1101	ddtowd	W64 = D64	conv

B	Group No.	OP. code	Dest. Reg.	Src. Reg.	not used
*	0 0 1 1	* * * *	* * * *	* * * *	

Group4 (Integer Arithmetic)

B	OP	Mnemonic	Operation	FU
*	0000	add	GR = GR + GR	ialu
*	0001	add	GR = GR + Signed Imm.13	ialu
*	0010	addc	GR = GR + GR + Cy	ialu
*	0011	addc	GR = GR + Signed Imm.13 + Cy	ialu
*	0100	sub	GR = GR − GR	ialu
*	0101	sub	GR = GR − Signed Imm.13	ialu
*	0110	subc	GR = GR − GR − Cy	ialu
*	0111	subc	GR = GR − Signed Imm.13 −Cy	ialu
*	1000	and	GR = GR & GR	ialu
*	1001	and	GR = GR & Signed Imm.13	ialu
*	1010	or	GR = GR | GR	ialu
*	1011	or	GR = GR | Signed Imm.13	ialu
*	1100	xor	GR = GR ⊗ GR	ialu
*	1101	xor	GR = GR ⊗ Signed Imm.13	ialu
*	1110	xnor	GR = ¬(GR ⊗ GR)	ialu
*	1111	xnor	GR = ¬(GR ⊗ Signed Imm.13)	ialu

B	Group No.	OP. code	Dest. Reg.	Src. Reg.	not used	Src. Reg.
*	0 1 0 0	* * * 0	* * * * *	* * * * *		* * * * *

B	Group No.	OP. code	Dest. Reg.	Src. Reg.	13bit Signed Immediate
*	0 1 0 0	* * * 1	* * * * *	* * * * *	

Group5 (Integer Arithmetic, Bit Operation)

B	OP	Mnemonic	Operation	FU
*	0000	extbit	GR = (GR & (1 ≪ GR)) ? 1 : 0	ialu
*	0001	extbit	GR = (GR & (1 ≪ Imm.5)) ? 1 : 0	ialu
*	0010	setbit	GR = GR | (1 ≪ GR)	ialu
*	0011	setbit	GR = GR | (1 ≪ Imm.5)	ialu
*	0100	resbit	GR = GR & ¬(1 ≪ GR)	ialu
*	0101	resbit	GR = GR & ¬(1 ≪ Imm.5)	ialu
*	1000			ialu
*	1001	neg	GR = (−GR)	ialu
*	1010			ialu
*	1011	prior	GR = priority(GR)	ialu
*	1100			ialu

B	Group No.	OP. code	Dest. Reg.	Src. Reg.	not used	Src. Reg.
*	0 1 0 1	* * * 0	* * * * *	* * * * *		* * * * *

B	Group No.	OP. code	Dest. Reg.	Src. Reg.	13bit Signed Immediate
*	0 1 0 1	* * * 1	* * * * *	* * * * *	

Group6(shift, rotate

B	OP	Mnemonic	Operation	FU
*	0000	shiftz	GR = GR ≪ GR or GR = GR ≫ GR	shift
*	0001	shiftz	GR = GR ≪ Imm.5 or GR = GR ≫ Imm.5	shift
*	0010	shifts	GR = GR ≪ GR or GR = GR ≫ GR	shift
*	0011	shifts	GR = GR ≪ Imm.5 or GR = GR ≫ Imm.5	shift
*	0100	rot	GR = GR rotate GR	shift
*	0101	rot	GR = GR rotate Signed Imm.6	shift
*	0111	slbz	GR = GR ≪ 1	shift
*	1001	slbc	GR = GR ≪ 1	shift
*	1011	srbz	GR = GR ≫ 1	shift
*	1101	srbc	GR = GR ≫ 1	shift
*	1111	srbs	GR = GR ≫ 1	shift

B	Group No.	OP. code	Dest. Reg.	Src. Reg.	not used	Src. Reg.
*	0 1 1 0	* * * 0	* * * * *	* * * * *		* * * * *

B	Group No.	OP. code	Dest. Reg.	Src. Reg.	13bit Signed Immediate
*	0 1 1 0	* * * 1	* * * * *	* * * * *	

Group7(Integer Multiply)

B	OP	Mnemonic	Operation	FU
*	0000	muls	GR = GR × GR	imul
*	0001	muls	GR = GR × Signed Imm.13	imul
*	0010	mulu	GR = GR × GR	imul
*	0011	mulu	GR = GR × Unsigned Imm.13	imul

B	Group No.			OP. code				Dest. Reg.					Src. Reg.					not used								Src. Reg.				
*	0	1	1	1	*	*	*	0	*	*	*	*	*	*	*	*	*	*								*	*	*	*	*

B	Group No.			OP. code				Dest. Reg.					Src. Reg.					13bit Signed Immediate												
*	0	1	1	1	*	*	*	1	*	*	*	*	*	*	*	*	*	*												

Group8(Floating-point Add, Subtract)

B	OP	Mnemonic	Operation	FU
*	0000	fadd.s	FR.s = FR.s + FR.s	falu
*	0001	fadd.d	FR.d = FR.d + FR.d	falu
*	0010	fadd.s	FR.s = |FR.s| + |FR.s|	falu
*	0011	fadd.d	FR.d = |FR.d| + |FR.d|	falu
*	0100	fadd.s	FR.s = |FR.s + FR.s|	falu
*	0101	fadd.d	FR.d = |FR.d + FR.d|	falu
*	1000	fsub.s	FR.s = FR.s − FR.s	falu
*	1001	fsub.d	FR.d = FR.d − FR.d	falu
*	1010	fsub.s	FR.s = |FR.s + FR.s|	falu
*	1011	fsub.d	FR.d = |FR.d + FR.d|	falu

B	Group No.			OP. code				Dest. Reg.					Src. Reg.					not used								Src. Reg.				
*	1	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*								*	*	*	*	*

Group9(Floating-point Multiply, Divide)

B	OP	Mnemonic	Operation	FU
*	0000	fmul.s	FR.s = FR.s × FR.s	fmul
*	0001	fmul.d	FR.d = FR.d × FR.d	fmul
*	0010	fmul.s	FR.s = |FR.s| × |FR.s|	fmul
*	0011	fmul.d	FR.d = |FR.d| × |FR.d|	fmul
*	0100	fmul.s	FR.s = FR.d × FR.d	fmul
*	0101	fmul.d	FR.d = FR.s × FR.s	fmul
*	1000	fdiv.s	FR.s = FR.s / FR.s	fdiv
*	1001	fdiv.d	FR.d = FR.d / FR.d	fdiv
*	1010	fdiv.s	FR.s = |FR.s / FR.s|	fdiv
*	1011	fdiv.d	FR.d = |FR.d / FR.d|	fdiv
*	1100	fdiv.s	FR.d = FR.s / FR.d	fdiv
*	1101	fdiv.d	FR.s = FR.s / FR.s	fdiv

B	Group No.			OP. code				Dest. Reg.					Src. Reg.					not used								Src. Reg.				
*	1	0	0	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*								*	*	*	*	*

Group10(18 bits Immediate)

B	OP	Mnemonic	Operation	FU
*	0000	ldil	GR = Signed Imm.18	shift
*	0001	ldih	GR = Signed Imm.18 ≪ 14	shift
*	0100	oril	GR = GR | Signed Imm.18	ialu
*	0101	orih	GR = GR | (Signed Imm.18 ≪ 14)	ialu
*	1000	fldil.s	FR.s = itofs(Signed Imm.18)	conv
*	1001	fldil.d	FR.d = itofd(Signed Imm.18)	conv
*	1010	fldih.s	FR.s = itofs(Signed Imm.18 ≪ 14)	conv
*	1011	fldih.d	FR.d = itofd(Signed Imm.18 ≪ 14)	conv

B	Group No.			OP. code				Dest. Reg.					18bit Signed Immediate																		
*	1	0	1	0	*	*	*	*	*	*	*	*	*																	

Group12 (Load)

B	OP	Mnemonic	Operation	FU
*	0000	ldub.so	GR = Unsigned memory(GR + Signed Imm.13)	load/store
*	0001	ldsb.so	GR = Signed memory(GR + Signed Imm.13)	load/store
*	0010	lduh.so	GR = Unsigned memory(GR + Signed Imm.13)	load/store
*	0011	ldsh.so	GR = Signed memory(GR + Signed Imm.13)	load/store
*	0100	ldw.so	GR = memory(GR + Signed Imm.13)	load/store
*	1000	ldub.wo	GR = Unsigned memory(GR + Signed Imm.13)	load/store
*	1001	ldsb.wo	GR = Signed memory(GR + Signed Imm.13)	load/store
*	1010	lduh.wo	GR = Unsigned memory(GR + Signed Imm.13)	load/store
*	1011	ldsh.wo	GR = Signed memory(GR + Signed Imm.13)	load/store
*	1100	ldw.wo	GR = memory(GR + Signed Imm.13)	load/store

B	Group No.		OP. code				Dest. Reg.					Src. Reg.					13bit Displacemenet												
*	1	1	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*											

Group13 (Load)

B	OP	Mnemonic	Operation	FU
*	0000	ldub.uo	GR = Unsigned memory(GR + Signed Imm.13)	load/store
*	0001	ldsb.uo	GR = Signed memory(GR + Signed Imm.13)	load/store
*	0010	lduh.uo	GR = Unsigned memory(GR + Signed Imm.13)	load/store
*	0011	ldsh.uo	GR = Signed memory(GR + Signed Imm.13)	load/store
*	0100	ldw.uo	GR = memory(GR + Signed Imm.13)	load/store
*	1000	flds.so	FR.d = memory(GR + Signed Imm.13)	load/store
*	1001	fldd.so	FR.d = memory(GR + Signed Imm.13)	load/store
*	1010	flds.wo	FR.d = memory(GR + Signed Imm.13)	load/store
*	1011	fldd.wo	FR.d = memory(GR + Signed Imm.13)	load/store
*	1100	flds.uo	FR.d = memory(GR + Signed Imm.13)	load/store
*	1101	fldd.uo	FR.d = memory(GR + Signed Imm.13)	load/store

B	Group No.		OP. code				Dest. Reg.					Src. Reg.					13bit Displacemenet												
*	1	1	0	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*											

Group14(Store)

B	OP	Mnemonic	Operation	FU
0	0000	stb.so	memory(GR + Signed Imm.13) = GR	load/store
0	0001	sth.so	memory(GR + Signed Imm.13) = GR	load/store
0	0010	stw.so	memory(GR + Signed Imm.13) = GR	load/store
0	0011	fsts.so	memory(GR + Signed Imm.13) = FR.s	load/store
0	0100	fstd.so	memory(GR + Signed Imm.13) = FR.d	load/store
0	0101	stb.wo	memory(GR + Signed Imm.13) = GR	load/store
0	0110	sth.wo	memory(GR + Signed Imm.13) = GR	load/store
0	0111	stw.wo	memory(GR + Signed Imm.13) = GR	load/store
0	1000	fsts.wo	memory(GR + Signed Imm.13) = FR.s	load/store
0	1001	fstd.wo	memory(GR + Signed Imm.13) = FR.d	load/store
0	1010	stb.uo	memory(GR + Signed Imm.13) = GR	load/store
0	1011	sth.uo	memory(GR + Signed Imm.13) = GR	load/store
0	1100	stw.uo	memory(GR + Signed Imm.13) = GR	load/store
0	1101	fsts.uo	memory(GR + Signed Imm.13) = FR.s	load/store
0	1110	fstd.uo	memory(GR + Signed Imm.13) = FR.d	load/store
0	1111			

B	Group No.				OP. code				Dest. Reg.					Src. Reg.					13bit Displacemenet												
0	1	1	1	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*											

付録 C

業績一覧

論文

1. Kazuaki MURAKAMI, Naohiko IRIE, Shinji TOMITA : "SIMP(Single Instruction stream/Multiple instruction Pipelining) : A Novel High-Speed Single Processor Architecture," *IEEE-CS & ACM : The 16th Annual International Symposium on Computer Architecture*, May 1989.
2. 久我守弘, 村上和彰, 富田眞治, "SIMP(単一命令流/多重命令パイプライン) 方式に基づく『新風』プロセッサの低レベル並列処理アルゴリズム," 情報処理学会並列処理シンポジウム'89, 1989 年 2 月.
3. 久我守弘, 入江直彦, 弘中哲夫, 村上和彰, 富田眞治 : "SIMP(単一命令流/多重命令パイプライン) 方式に基づく『新風』プロセッサの低レベル並列処理アルゴリズム," 情報処理学会論文誌, 第 30 巻, 第 12 号, 1989 年 12 月.
4. 久我守弘, 入江直彦, 村上和彰, 富田眞治 : "SIMP(単一命令流/多重命令パイプライン) 方式に基づくスーパースカラ・プロセッサ『新風』の性能評価," 情報処理学会並列処理シンポジウム'90, 1990 年 5 月.
5. 納富昭, 原哲也, 久我守弘, 村上和彰, 富田眞治 : "DSN 型スーパースカラ・プロセッサ・プロトタイプ —アーキテクチャおよび静的コード・スケジューリングに関する総合評価—," 情報処理学会並列処理シンポジウム'91, 1991 年 5 月.
6. Morihiro KUGA, Kazuaki MURAKAMI, Shinji TOMITA : "DSNS(Dynamically-hazard-resollved, Statically-code-scheduled, Nonuniform Superscalar) : Yet Another Superscalar Processor Architecture," *ACM SIGARCH Computer Architecture News*, Vol.19, No.4, pp.14–29, June 1991.

7. 久我守弘, 入江直彦, 村上和彰, 富田眞治 : "SIMP(単一命令流/多重命令パイプライン) 方式に基づくスーパースカラ・プロセッサ『新風』の性能評価," 情報処理学会論文誌, 第 32 巻, 第 7 号, 1991 年 7 月.

講演論文

1. 村上和彰, 久我守弘, 富田眞治 : "『新風』DTS-edition : SIMP(単一命令流/多重命令パイプライン) 方式に基づくデスクトップ・スーパーコンピュータ," 情報処理学会第 37 回全国大会, 4N–1, 1988 年 9 月.
2. 久我守弘, 村上和彰, 富田眞治 : "『新風』プロセッサの高速化メカニズム," 情報処理学会第 37 回全国大会, 4N–2, 1988 年 9 月.
3. 入江直彦, 久我守弘, 村上和彰, 富田眞治 : "SIMP(単一命令流/多重命令パイプライン) 方式のシミュレーションによる評価," 情報処理学会第 37 回全国大会, 4N–3, 1988 年 9 月.
4. 久我守弘, 村上和彰, 富田眞治 : "『新風』プロセッサの命令実行制御," 第 41 回電気関係学会九州支部連合大会, 925, 1988 年 10 月.
5. 入江直彦, 久我守弘, 村上和彰, 富田 眞治 : "SIMP(単一命令流/多重命令パイプライン) 方式に基づく『新風』プロセッサの高速化技法および性能予測," 情報処理学会計算機アーキテクチャ研究会, ARC–73–11, 1988 年 11 月.
6. 久我守弘, 村上和彰, 富田眞治 : "『新風』プロセッサの命令セット・アーキテクチャ," 情報処理学会第 38 回全国大会, 5T–8, 1989 年 3 月.
7. 弘中哲夫, 久我守弘, 村上和彰, 富田眞治 : "ストリーム FIFO 方式に基づくベクトル・プロセッサ『順風』," 電子情報通信学会コンピュータシステム研究会, CPSY89–39, 1989 年 8 月.
8. 原哲也, 久我守弘, 村上和彰, 富田眞治 : "『新風』プロセッサにおける分岐予測," 情報処理学会第 39 回全国大会, 6X–5, 1989 年 10 月.
9. 入江直彦, 久我守弘, 村上和彰, 富田眞治 : "『新風』プロセッサのための静的コードスケジューリング," 情報処理学会第 39 回全国大会, 6X–6, 1989 年 10 月.
10. 弘中哲夫, 久我守弘, 村上和彰, 富田眞治 : "ストリーム FIFO 方式に基づくベクトル・プロセッサ『順風』の構成," 情報処理学会第 39 回全国大会, 7X–3, 1989 年 10 月.

11. 入江直彦, 久我守弘, 村上和彰, 富田眞治 : “SIMP(単一命令流/多重命令パイプライン) 方式に基づくスーパースカラ・プロセッサ『新風』のための静的コード・スケジューリング技法,” 情報処理学会計算機アーキテクチャ研究会, ARC-79-6, 1989 年 11 月.

12. 原哲也, 久我守弘, 村上和彰, 富田眞治 : “SIMP(単一命令流/多重命令パイプライン) 方式に基づくスーパースカラ・プロセッサ『新風』の命令供給機構,” 情報処理学会計算機アーキテクチャ研究会, ARC-80-7, 1990 年 1 月.

13. 弘中哲夫, 岡崎恵三, 久我守弘, 村上和彰, 富田眞治 : “ストリーム FIFO 方式に基づくベクトル・プロセッサ『順風』 —IF 文を含む DO ループの処理—,” 情報処理学会第 40 回全国大会, 7L-4, 1990 年 3 月.

14. 久我守弘, 納富昭, 村上和彰, 富田眞治 : “『新風』プロセッサの依存解析機能付きレジスタファイル,” 情報処理学会第 40 回全国大会, 7L-5, 1990 年 3 月.

15. 原哲也, 久我守弘, 村上和彰, 富田眞治 : “『新風』プロセッサの条件分岐方式,” 情報処理学会第 40 回全国大会, 7L-6, 1990 年 3 月.

16. 村上和彰, 久我守弘, 富田眞治 : “SIMP(単一命令流/多重命令パイプライン) 方式に基づくスーパースカラ・プロセッサの改良方針,” 電子情報通信学会コンピュータシステム研究会, CPSY90-54, 1990 年 7 月.

17. 原哲也, 納富昭, 久我守弘, 村上和彰, 富田眞治 : “SIMP(単一命令流/多重命令パイプライン) 方式に基づく改良版スーパースカラ・プロセッサの構成と処理,” 電子情報通信学会コンピュータシステム研究会, CPSY90-55, 1990 年 7 月.

18. 納富昭, 久我守弘, 村上和彰, 富田眞治 : “『新風』プロセッサのマルチポート・データキャッシュ,” 情報処理学会第 41 回全国大会, 3P-1, 1990 年 9 月.

19. 原哲也, 久我守弘, 村上和彰, 富田眞治 : “DSN 型スーパースカラ・プロセッサ・プロトタイプの分岐パイプライン,” 情報処理学会計算機アーキテクチャ研究会, ARC-86-3, 1991 年 1 月.

20. 納富昭, 久我守弘, 村上和彰, 富田眞治 : “DSN 型スーパースカラ・プロセッサ・プロトタイプのロード/ストア・パイプライン,” 情報処理学会計算機アーキテクチャ研究会, ARC-86-4, 1991 年 1 月.