1B-3

# 連想プロセッサDREAM-Iのマイクロアセンブラ(DRMASS)

上森 明　田中英彦　元岡 達
(東京大学 工学部)

## 1. はじめに

パターン処理、データベース管理等への応用を目的とした連想プロセッサシステムDREAM-I用のマイクロアセンブラDRMASS(DReaM-I's ASSembler)を紹介する。このアセンブラの特徴は、

- レジスタ、演算回路間のデータフローを式の形で記述
- 並列に動作可能な機能は、複合代入文で記述
- 連想プロセッサ内のレジスタと種々の演算回路との許される組合せを全て記述可能であり、許されない組合せはエラーとする

等の点にある。つまり、DREAM-Iに用意された演算機能を十分に利用できるように、記述形式に工夫をして記述の容易さを目ざしている。

## 2. データ処理の表現

DREAM-Iには、通常のALU演算の他に、外部機能として、二次元アクセス、バレルシフト、ビットグループ交換、バブルロジック、プライオリティエンコーダの5種類の演算回路がある。これらの外部機能とマイクロプロセッサ内のALUとは、組合せて動作させる事ができる。例えば、DRMASSの記法で、

SE(<regA>) AND Q=<regB>;

は、<regA>をバレルシフトした後、マイクロプロセッサ内のQレジスタで論理積をとり、<regB>へ格納する。等号は、データが左辺から右辺へ流れる事を意味する。外部機能は、関数の形で記述される。<regA>,<regB>はマイクロプロセッサ内のレジスタファイルである。

又、マイクロプロセッサ内部での演算と、外部レジスタ間の転送は、同時に動作可能で、DRMASSでは、

<regA> ADD <regB>=<regB>,
IO2=TDAM;

のような記述方法をとっている。

更に、ALUの演算結果は、内部のレジスタファイルと、外部レジスタとに、同時に格納できる。これは、例えば、

<regA> SUB <regB>=<regB>=TDAM;

のように記述され、<regB>と二次元記憶に同時に演算結果が格納される。

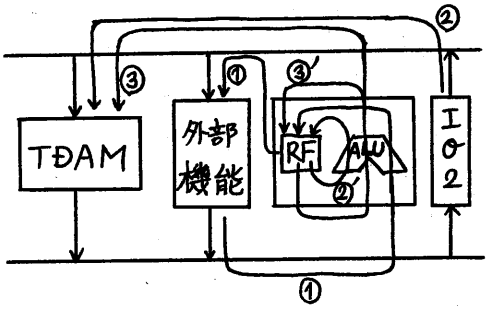


図1. DREAM-I内で許されるデータフローの例

図1は、上記の3つの例をまとめたものである。

DRMASSでは、DREAM-Iで用意された各機能の許される組合せを全て記述できるようにした。逆に、許されない組合せはチェックするようにし、冗長な論理機能や意味のない演算は、文法仕様から はずした。

DRMASSが、通常のニモニックによる表現を用いずに、式やデータフローの形の表現を用いた理由として、以下の点があげられる。

- ALUと外部演算回路が並列に動

作可能なので、両方の動作指定が必要な事
- 内部レジスタ、外部レジスタへの同時代入が可能な事
- DREAM-エ内のレジスタトランスファレベルのデータフローをわかり易く記述したい事

、等である。

## 3. まとめ

DRMASSは、FORTRANで記述され、約1000行である。現在、DRMASSを用いて、種々の応用プログラムを開発中であり、DREAM-エの評価を行なっている。なお、以下に、DRMASSの文法仕様を示した。

[参考文献] 元岡、田中、上森、河村「連想プロセッサのハードウェア構成と処理機能」信学技報 EC77-77

[付録] DRMASSの文法仕様

1、擬似命令
- ORG {<imm8> / <label>};
- [<label>:] EQU {<imm8> / <label>};
- END;
- <imm8> ::= 0～255 | #00～#FF

2、分岐命令
- [<label>:] {NOP | POP | PUSH | RTS};
- [<label>:] EOL [{(Z) / (C)}];
- [<label>:] BR (D);
- [<label>:] BR [(NZ) | (Z) | (N) | (V) | (C)] {<imm8> / <label>};
- [<label>:] JSR [(NZ)] {<imm8> / <label>};

3、直接データ命令
- [<label>:] {<imm16> / <label>} = <reg B>;
- <imm16> ::= 0～65535 | #0000～#FFFF

4、内部演算命令
- [<label>:] <ALU expression> = {X | Q | <reg B> [(<shift mode>)]
  [, <reg A>] = {IO1 / TDAM / TMAR}
  [{ , TDAM = IO2 / , IO2 = TDAM }];
- <ALU expression> ::= ZERO | <unary term expression> | <binary term expression>
- <unary term expression> ::= <f1> <reg A> | <f1> <reg B> | <f1> <reg D> | <f1> Q
- <f1> ::= INV | PASS | INC | DEC | NEG
- <binary term expression> ::= <reg A> <f2> <reg B> | <reg B> <f2> <reg A> | <reg A> <f2> <reg D> | <reg D> <f2> <reg A> | <reg A> <f2> Q | Q <f2> <reg A> | <reg D> <f2> Q | Q <f2> <reg D>
- <reg D> ::= IO1 | TDAM
- <reg A> ::= R0～R15 | R#0～R#F
- <reg B> ::= R0～R15 | R#0～R#F
- <shift mode> ::= ROL | ROR | DROL | DROR | DASR | DASL

5、外部機能命令
- [<label>:] <external logic expression> = <reg B>;
- <external logic expression> ::= <external function> | <unary ext func> | <binary ext func> | <SEM inst>
- <external function> ::= SE(<reg A>) | PE(<reg A>) | BL(<reg A>)
- <unary ext func> ::= <f1> <external function>
- <binary ext func> ::= <external function> <f2> {Q / <reg A>} | {Q / <reg A>} <f2> <external function>
- <SEM inst> ::= SE(<reg A>, {BS / EX}, <imm4>
- <imm4> ::= 0～15 | #0～#F

6、外部定義命令
- [<label>:] XOP;