Manchester Dataflow Processor

Ana Caroline Spengler – 8532356
Gil Barbosa Reis – 8532248
Paulo Bardes Nogueira Nascimento – 8531932

***Resumo*—Nesse documento discutiremos as características e implementação da a máquina Dataflow de Manchester, uma máquina dataflow *tagged token* (dinâmica) criada no início dos anos 80.**

***Index Terms*—Manchester, Dataflow, Arquitetura**

I. Introdução

EM meados dos anos 70 e 80 maquinas dataflow cresceram em popularidade na área acadêmica, tornando-se alvo de diversos estudos. Múltiplos projetos e protótipos começaram a surgir ao redor do mundo para demonstrar sua aplicabilidade e desempenho. Entre essas máquinas estava o projeto da Máquina Dataflow de Manchester.

Poucos projetos chegaram a de fato implementar os designs propostos, e meno ainda atingiram um protótipo funcional. A da Máquina de Manchester foi um dos poucos projetos que saiu do papel, não só um protótipo funcional, mas com um *toolkit* bastante completo incluindo: compilador, assembler e simulador.

Sendo uma máquina baseada no fluxo de dados. Sendo assim, instruções não têm posições de memória específicas, podendo serem concebidas como operações puras. Ela é uma máquina dataflow dinâmica, ou seja, os dados são transmitidos em pacotes rotulados, chamados *Tokens*.

II. Dataflow

A. Modelo dataflow

Dataflow é um paradigma de computação baseado no fluxo de dados, ou seja, a sequência de instruções necessárias para o processamento dos dados não é definida pelo programador. Seu trabalho é elaborar uma rede de conexões entre operações básicas, de modo que os dados possam fluir pelas conexões, sendo computados concorrentemente.

Como os dados fluem pelos caminhos, não há variáveis ou posições de memória que possam referenciá-los. É possível porém, definir um valor inicial para garantir o conteúdo durante a execução inicial do procedimento.

Além disso, os dados só podem fluir pelas unidades funcionais se todos os operadores necessários estiverem disponíveis. Caso contrário, os dados ficam parados até que os outros estejam disponíveis.

B. A estrutura de um programa dataflow

Assim como para sistemas computacionais convencionais, programas dataflow podem ser escritos em linguagens de programação de alto nível. O tipo mais conveniente de linguagem para compilar programas dataflow são linguagens com atribuição simples, ou seja, exceto pelo valor inicial, cada símbolo só pode ser atribuído uma vez. Isso é mostrado na Listagem 1, que define um programa na linguagem Sisal que calcula a área abaixo da curva $y = x^2$.

Listing 1: Programa exemplo em Sisal

```
export Integrate

function Integrate (returns real)

for initial
    int := 0.0;
    y   := 0.0;
    x   := 0.02
while
    x < 1.0
repeat
    int := 0.01 * (old y + y);
    y   := old x * old x;
    x   := old x + 0.02
returns
    value of sum int
end for

end function
```

A compilação de programas em alto nível são inicialmente traduzidas para uma linguagem de nível intermediário, quase equivalente ao *macroassembler* convencional, como a linguagem TASS, mostrado na Listagem 2.

A última fase da compilação forma um programa em código de máquina com a entrada. A Figura 2 mostra a representação gráfica usual de programas dataflow.

C. Dataflow dinâmico

Existem dois tipos de maquinas dataflow: **dinâmicas** e **estáticas**. Em máquinnas estáticas não é possível executar nenhum tipo de código reentrante sem auxilio de alguma extrutura interna para garantir que *tokens* de contextos diferentes não se misturem.

Já maquinas dinâmicas permitem a execução de códigos reentrantes usando-se rótulos (*tags*), que atuam como identificadores de contexto de cada token, permitindo diferenciar cada um deles dinâmicamente em tempo de execução e tratá-los corretamente.

III. Implementação

A. *Tokens*

A Máquina de Manchester possui *tokens* representados geralmente por 96 bits divididos da seguinte forma: dados (37 bits), rótulo (36 bits), destino (22 bits), marcador (1 bit).

O rótulo por sua vez e subdividido em três tipos:

Nível de Iteração
: Usado para diferenciar tokens de difernetes iterações de um loop.

Nome de Ativação
: Usado para diferenciar diferentes chamadas de um trecho de código, inclusive chamadas recursivas.

Índice
: Usado quando uma mesma operação é aplicada a diversos elementos de uma estrutura de dados.

A presença desses rótulos é essencial para permitir a concorrência de códigos reentrantes. Do contrário trechos repetidos de código teriam que ser executados de maneira síncrona, com apenas tokens de um mesmo contexto presente de cada vez, o que prejudicaria muito o desempenho e eficiência.

B. *Token Ring*

O *Token Ring* é um conjunto de unidades funcionais onde os tokens circulam, sendo modificados e processados. A Figura 1 mostra as unidades funcionais do token ring. Cada estágio pode alterar o token acrescentando ou removendo informações conforme o necessário.

Devido a limitações de largura de banda é essencial que o tamanho dos tokens seja o menor possível, por isso diferentes formatos de tokens existem para permitir maior flexibilidade. Por exemplo um token pode conter dois destinos para economizar um *DUP* ou ele pode conter uma entrada e um valor constante, para transformá-lo numa operação unária (mais rápida pois passa direto pela *Matching Unit*).

I/O Switch
: O I/O Switch é o bloco de entrada e saída da máquina. É usado para carregar o programa na memória e transferir resultados de volta.

Token Queue
: É uma estrutura que enfileira os pacotes, para que esses fluam com uma taxa compatível com a taxa de entrada de tokens na Matching Unit.

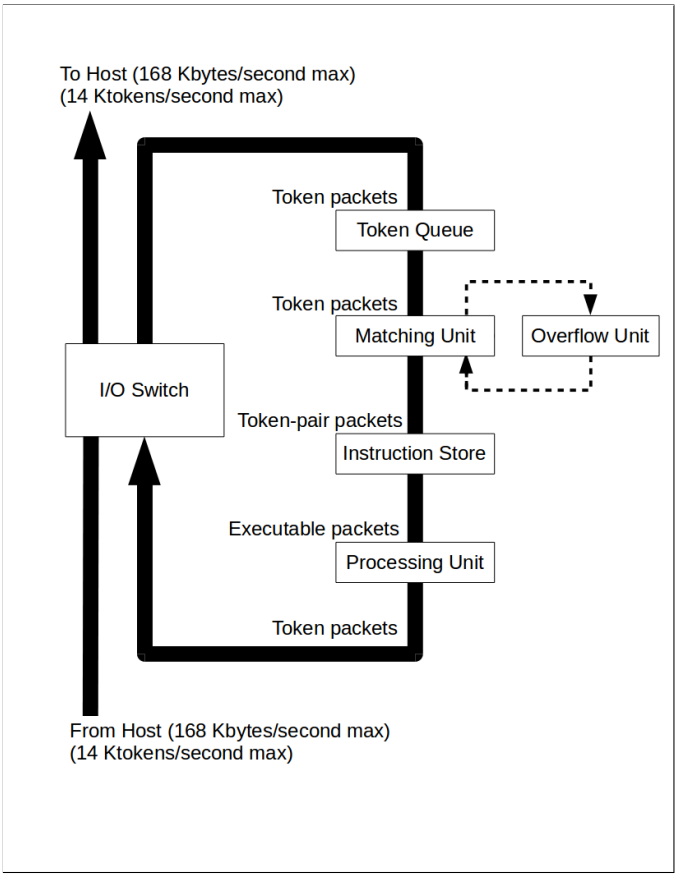
Matching Unit
: É a responsável por unir tokens de entrada de uma mesma operação. Essa combinação é feita levando-se em consideração os destinos e os rótulos. É usado um algorítimo de hash para isso. Quando todo o espaço da tabela hash está ocupado, tokens são armazenados na Overflow Unit.

Overflow Unit
: Faz o mesmo papel da Matching Unit, porém sacrificando performance em prol de maior espaço para tokens excedentes.

Instruction Store
: É uma estrutura que recebe tokens cominados e buscar o *opcode* da operação correspondente ao destino dos tokens recebidos, montando um pacote executável.

Processing Unit
: Unidade que recebe o pacote executável, o preprocessa e encaminha para a unidade funcional correspondente à operação. Após execução da operação, um novo token é gerado e mandado para o Token Queue, recomeçando o ciclo.

Figura 1: Token Ring da máquina dataflow de Manchester



Referências

[1] J. R. GURD, C. C. KIRKHAM, and I. WATSON (1985). *THE MANCHESTER PROTOTYPE DATAFLOW COMPUTER*. Communications of the ACM, 28 (1), 34-52.

Listing 2: Programa acima escrito em TASS

```
(\I "TASS" "TSM")

! Integration by trapezoidal rule
! ================================

! initialize the loop variables
int         = (Data "R 0.0");
y           = (Data "R 0.0");
x           = (Data "R 0.02");

! merge the initial values with the loop
! output values
int_mrg     = (Mer int new_int);
y_mrg       = (Mer y new_y);
x_mrg       = (Mer x new_x);

! test for termination of loop
test        = (CGR "R 1.0" x_mrg);

! gate the loop variables into new loop
! instance or direct result to output
gate_int    = (BRR int_mrg test);
old_int     = gate_int.R;
old_y       = (BRR y_mrg test).R;
old_x       = (BRR x_mrg test).R;

result      = (SIL gate_int.L "O 0").L;

! loop body: form new values for loop
! variables
incr_x      = (ADR old_x "R 0.02");
x_sq        = (MLR old_x old_x);
height_2    = (ADR old_y x_sq);
area        = (MLR "R 0.01" height_2);
cum_area    = (ADR old_int area);

! increment iteration level for new loop
! variables
new_int     = (ADL cum_area "I 1").L;
new_y       = (ADL x_sq "I 1").L;
new_x       = (ADL incr_x "I 1").L;

! output the final value of int
            (OPT result "G 0");
(Finish);
```

Figura 2: Fluxo de dados do Listing 1