Manchester Dataflow Processor

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

ICMC - USP São Carlos

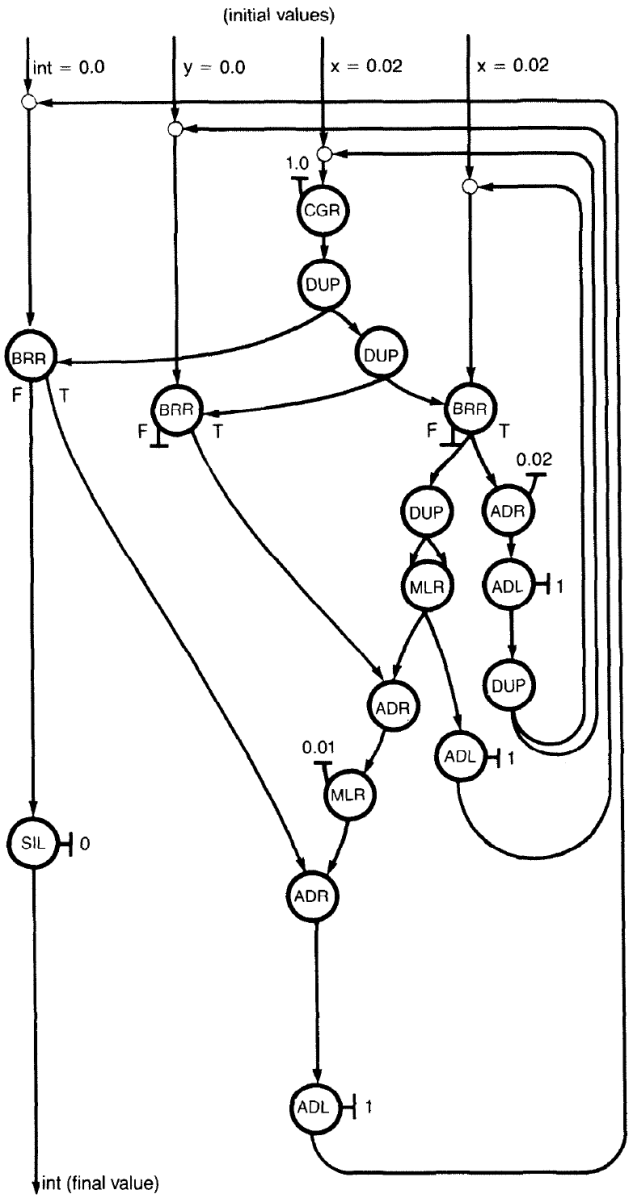
16 de novembro de 2015

Sumário

Dataflow

- Fluxo de dados - Rede de conexões entre operações básicas
- Não há variáveis, nem posições de memória para referência
- Dados fluem por unidades funcionais para computação
- Só há processamento de dados se todos operadores necessários estão disponíveis
- Dados são transmitidos em pacotes rotulados, chamados *Tokens*

Figura: Fluxo de dados do Programa 1



Máquina Dataflow de Manchester

- Dataflow Dinâmico
- Cada pacote é rotulado, identifcando o contexto de cada token, permitindo diferenciá-los em tempo de execução
- Permite execução de código reentrante

Token

Tokens de 96 bits

- Dados (37 bits)
- Rótulo (36 bits)

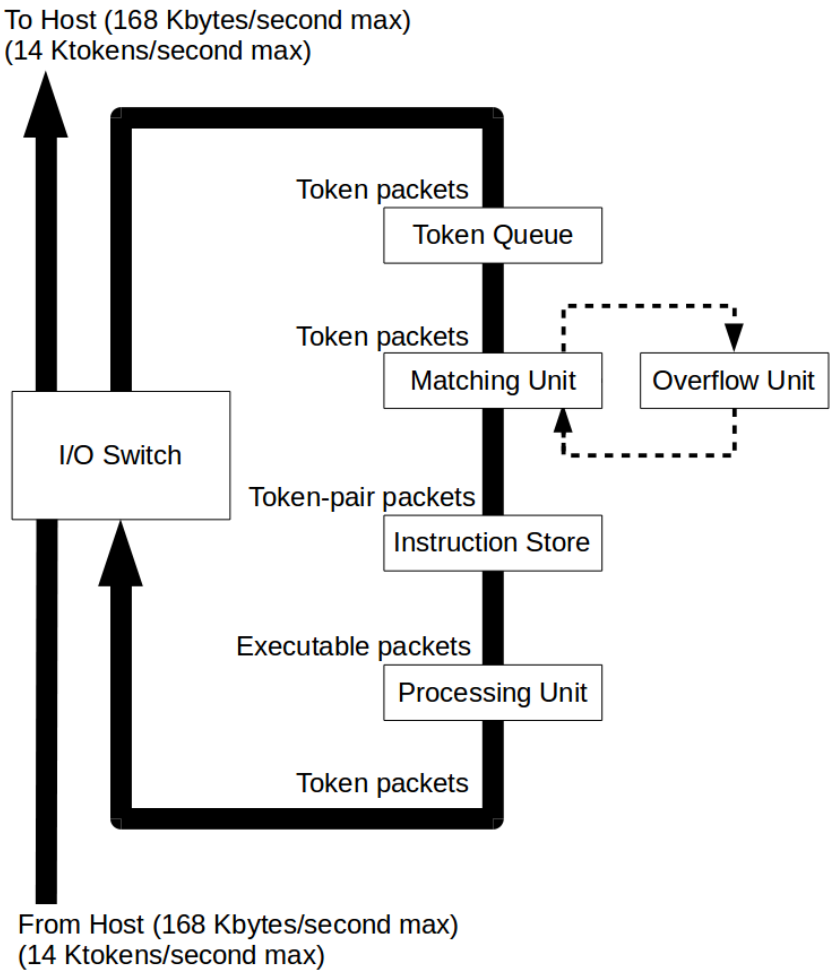
  Nível de Iteração Usado para diferenciar tokens de diferentes iterações de um loop

  Nome de Ativação Diferencia diferentes chamadas de um trecho de código, incluindo chamadas recursivas

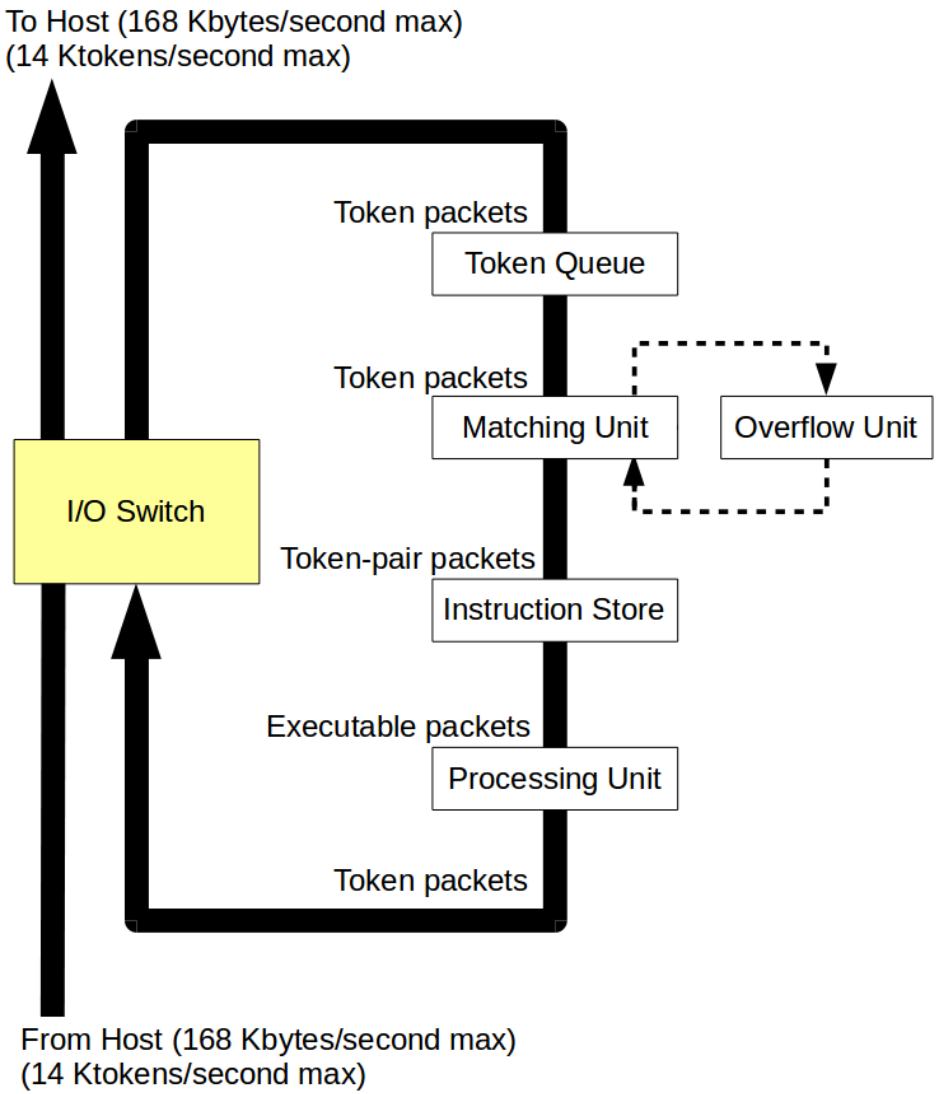
  Índice Usado quando uma mesma operação é aplicada a diversos elementos de uma estrutura de dados
- Destino (22 bits)
- Marcador (1 bit)

Token Ring

Figura: Token Ring da Máquina Dataflow de Manchester

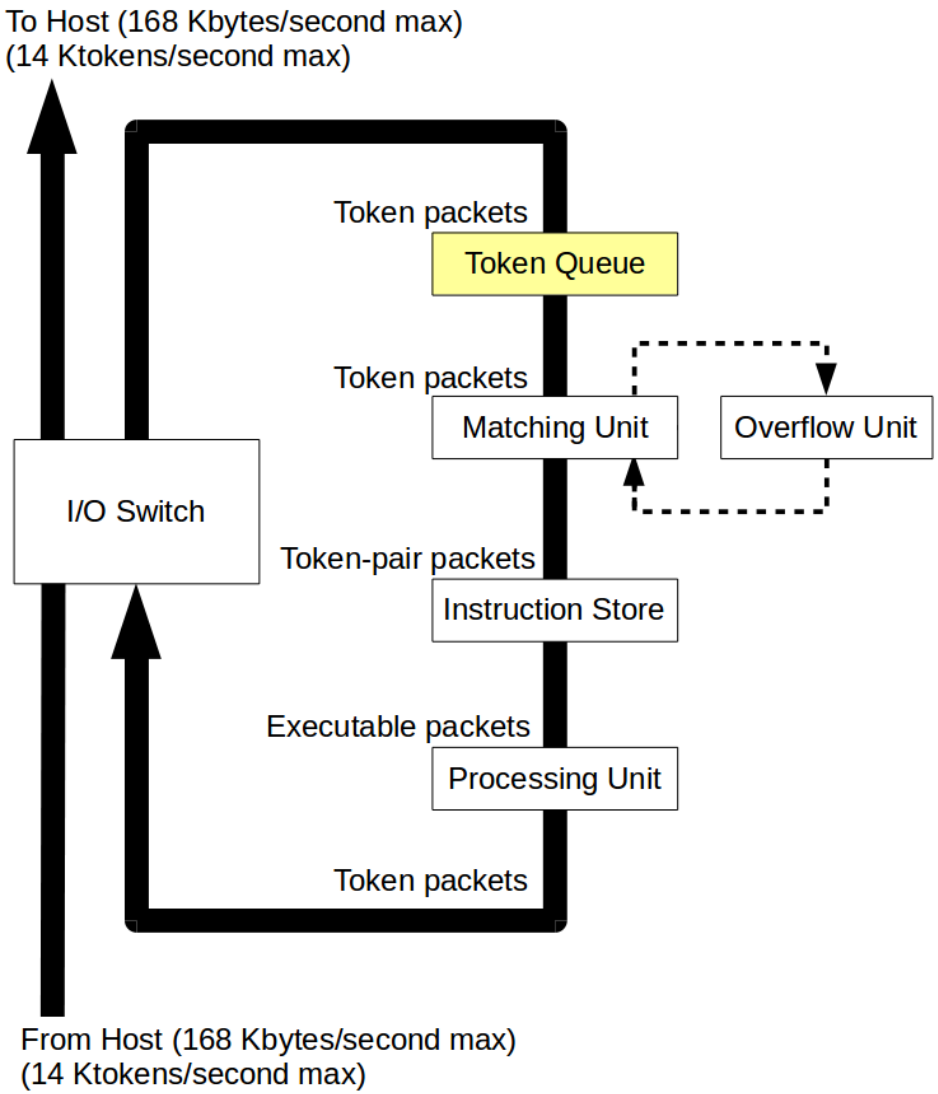


I/O Switch



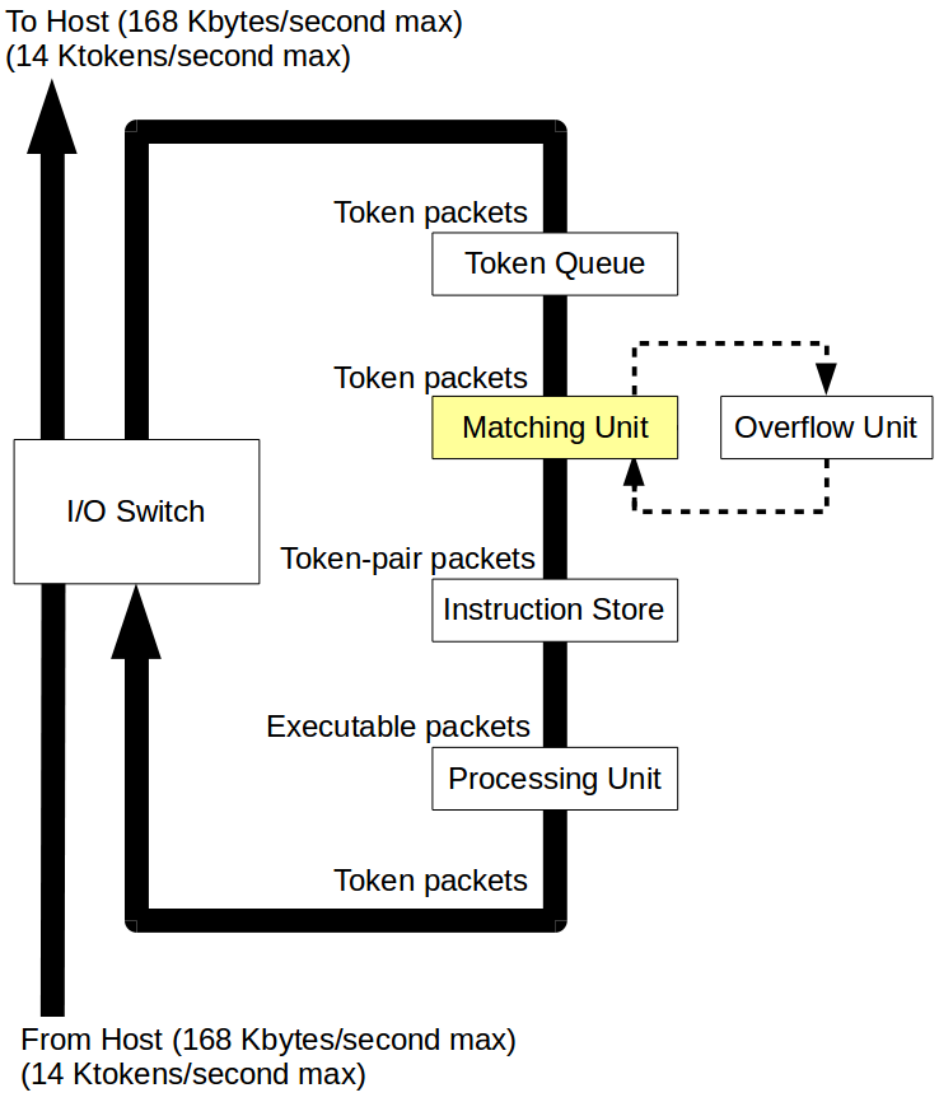
O I/O Switch é o bloco de entrada e saída da máquina. É usado para carregar o programa na memória e transferir resultados de volta.

Token Queue



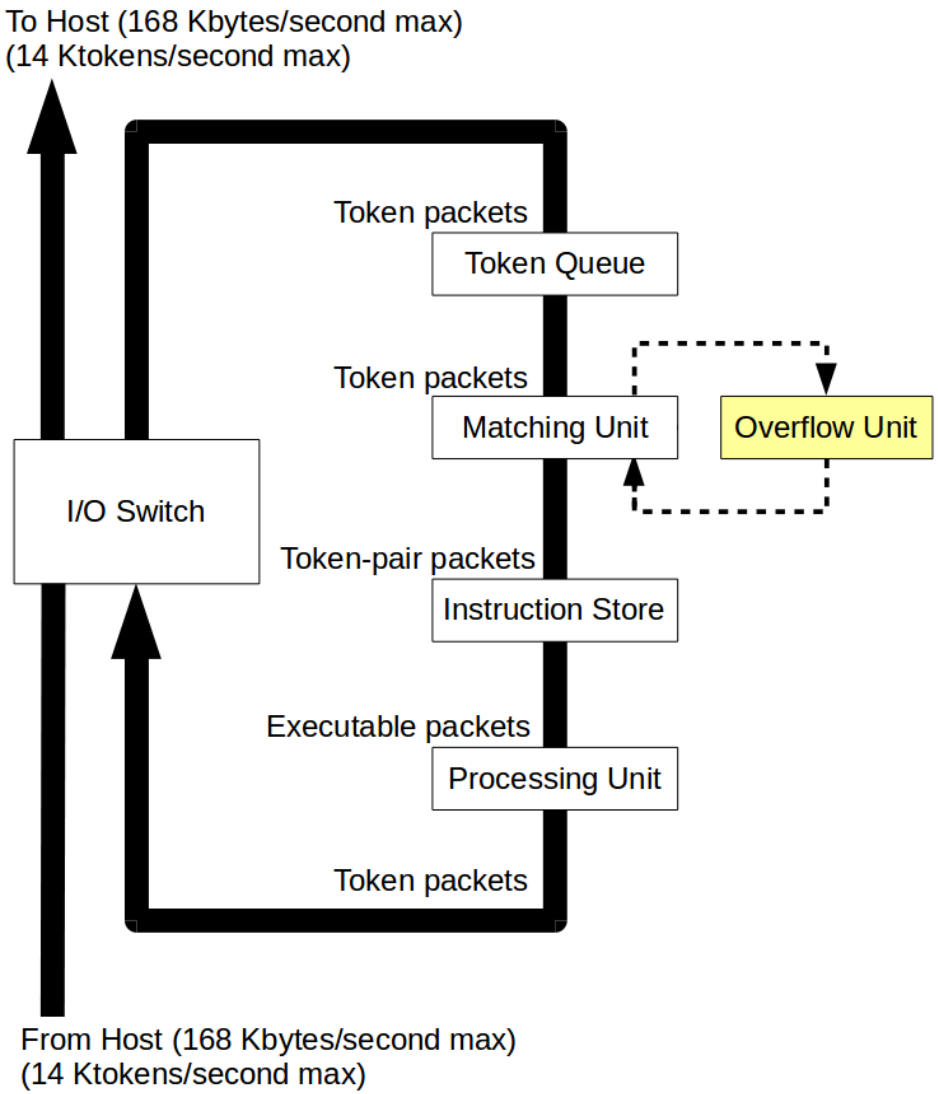
É uma estrutura que enfileira os pacotes, para que esses fluam com uma taxa compatível com a taxa de entrada de tokens na Matching Unit.

Matching Unit



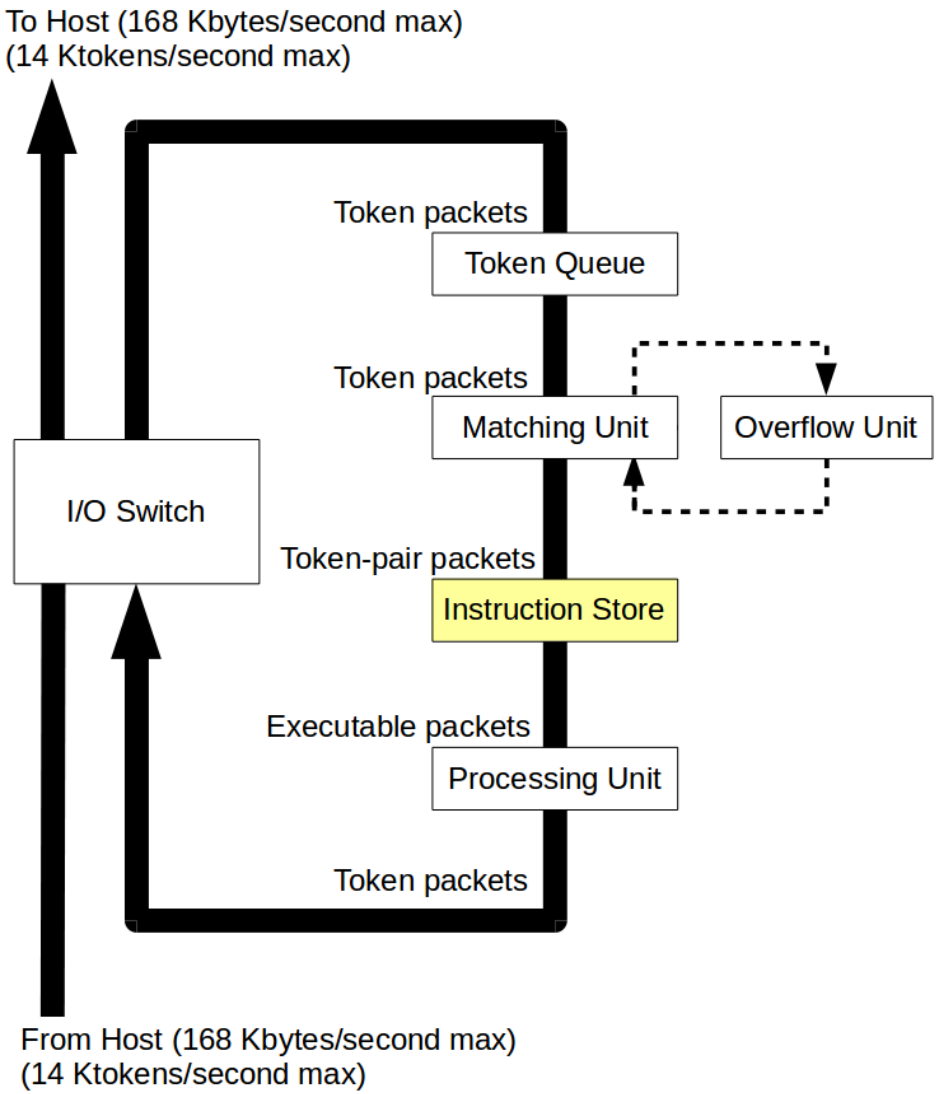
É a responsável por unir tokens de entrada de uma mesma operação. Essa combinação é feita levando-se em consideração os destinos e os rótulos. É usado um algorítimo de hash para isso. Quando todo o espaço da tabela hash está ocupado, tokens são armazenados na Overflow Unit.

Overflow Unit



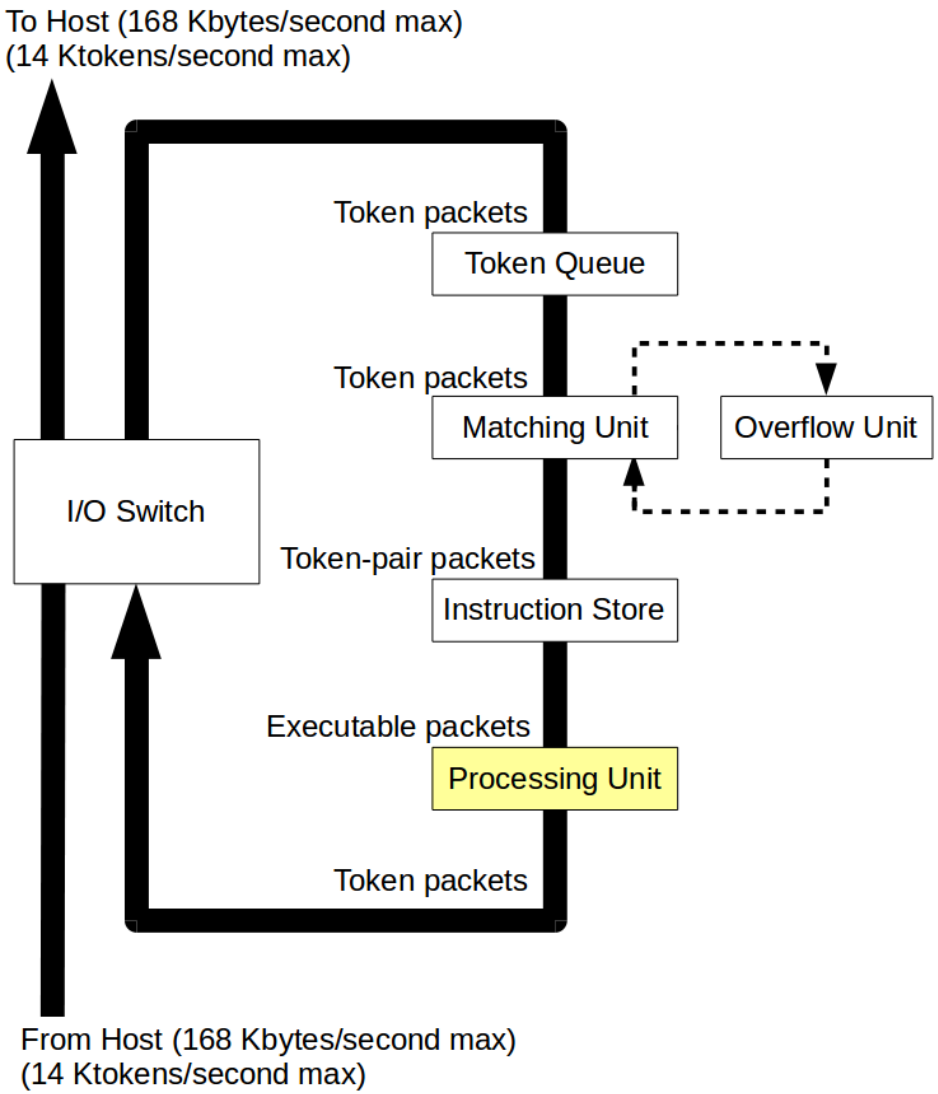
Faz o mesmo papel da Matching Unit, porém sacrificando performance em prol de maior espaço para tokens excedentes.

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 pre-processa 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.

Código Alto Nível

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
```

Código Nível Intermediário

Listing 2: Programa anterior 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 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);
```

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