

APS LOGCOMP

HAMBURGUERIAVM

LINGUAGEM DE PROGRAMAÇÃO TEMÁTICA + MÁQUINA VIRTUAL

MARIA VITORIA JARDIM SARTORI



MOTIVAÇÃO



- EXPLORAR O CICLO COMPLETO DE UM COMPILADOR REAL
- CRIAR UM AMBIENTE SIMULÁVEL SIMPLES, COM REGRAS CLARAS:
 - MESAS → PEDIDOS → FILA → ENTREGA → PAGAMENTO
- PROVAR QUE CONCEITOS DE COMPILADORES PODEM SER APLICADOS DE FORMA CRIATIVA



HAMBURGUERIAVM: O QUE ELA É?

- UMA LINGUAGEM DE ALTO NÍVEL QUE SIMULA O FUNCIONAMENTO DE UMA HAMBURGUERIA
- SINTAXE SIMPLES E INSPIRADA EM LINGUAGENS ESTRUTURADAS
- FOCADA NO CONTROLE DE:
 - MESAS,
 - PEDIDOS,
 - FILA GLOBAL,
 - ESTADO DE PAGAMENTO
- USA ESTRUTURAS CLÁSSICAS:
 - VARIÁVEIS (IDENTIFICADORES),
 - CONDICIONAIS (IF / ELSE),
 - LOOPS (WHILE).

PIPELINE

CÓDIGO-FONTE (EXEMPLO.HBURG)

ANÁLISE LÉXICA – FLEX

ANÁLISE SINTÁTICA – BISON

GERAÇÃO DE CÓDIGO (OUT.ASM) – VIA AÇÕES DO PARSER

VM HAMBURGUERIAVM – EXECUTA O ASSEMBLY

SAÍDA – MOSTRADA AO USUÁRIO (FILA, PEDIDOS, PRINTS, ENTREGAS)



EXEMPLO ALTO-NÍVEL



```
abrir_mesa();
fazer_pedido(mesa1, hambuguer);
print("Pedido registrado");
```

```
if mesa1 == aberta then {
    entregar_proximo;
} else {
    print("Mesa fechada");
}
```

```
while fila not empty {
    entregar_proximo;
}
```

```
pagar(mesa1);
```

EXEMPLO ASSEMBLY



```
OPEN_TABLE 1
ORDER mesa1 hamburguer
PRINT_STR "Pedido registrado"

JZ mesa_mesa1_aberta else_0
DELIVER
GOTO end_if_0
else_0:
PRINT_STR "Mesa fechada"
end_if_0:
```

```
while_start_1:
JZ fila_not_empty while_end_1
DELIVER
GOTO while_start_1
while_end_1:
PAY mesa1
HALT
```



HAMBURGUERIAVM (PYTHON)

- Simula um restaurante:
 - fila de pedidos (deque)
 - entregas
 - mesas e estados
- Instruções:
 - OPEN_TABLE, ORDER, PAY, DELIVER
 - Saltos: GOTO, JZ
 - Impressão: PRINT_STR
- Interpreta linha por linha
Comportamento determinístico