Implementação em Hardware de Instrução Segura de Acesso à Memória - Caso MIPS 16 bit

Eric S. Torres , Antonio L. Maia Neto, Omar P. Vilela Neto, Leonardo B. Oliveira

Departamento de Ciência da Computação Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG, Brasil

Introdução

- Segurança de Software
  - Integridade de dados
  - Confiabilidade de dados
- Linguagens
  - Focam eficiência
  - Mínima interferência na execução
  - Não possuem mecanismos de segurança
  - Ficam a cargo do desenvolvedor

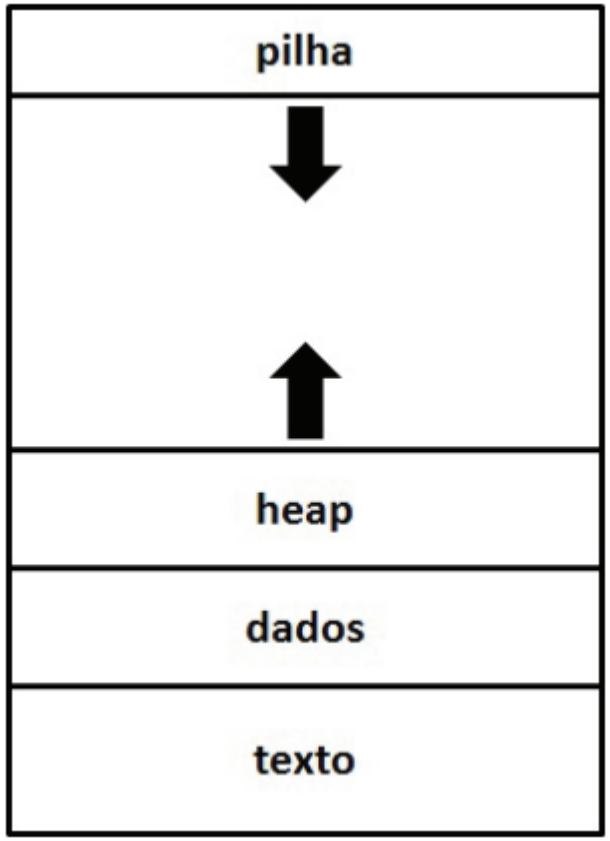
Introdução

- *Buffer Overflow* – BOF
  - Escrever mais dados do que o arranjo suporta
  - Worm Morris 1988
  - Execução DoS
- Verificação de Limites de Arranjo
  - *(Array Bound Check)*- ABC
  - Eficaz
  - Ineficiente

Segurança de Software

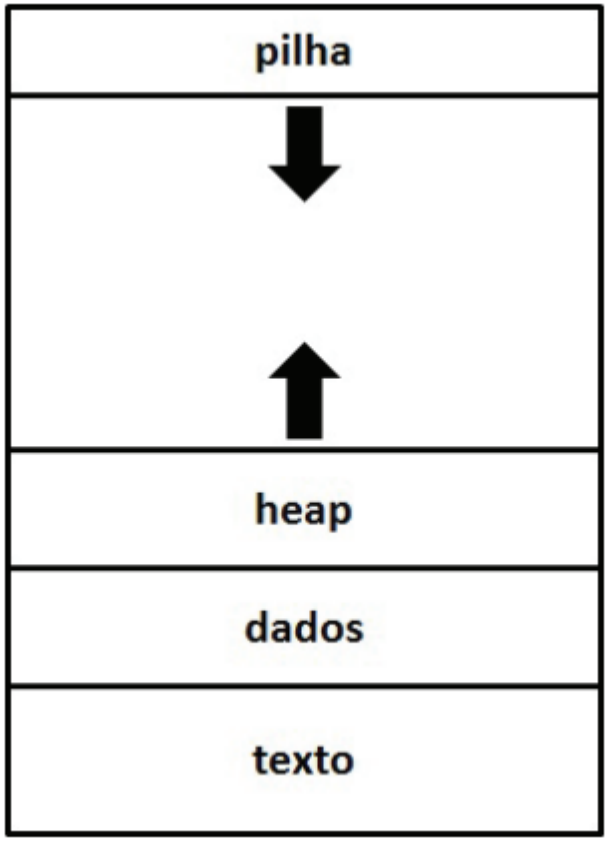
- Exploração de Vulnerabilidade
  - Sigilo
  - Integridade*
  - Disponibilidade
  - Autenticidade
- Proposta do trabalho
  - Acesso a memória seguro
  - Instrução de Hardware para ABC

Memória de um processo



- Texto - regiao de memória estática – contém o código
- Dados – estática - armazenadas constantes e variaveis globais.
- Heap – dinâmica - armazena porções de memória para o programa
- Pilha – dinâmica e contígua - armazena variáveis de controle e realiza troca de contexto entre procedimentos.

Memória de um processo



Ataques ocorrem de duas formas:

- Pilha
  - Sobreescrever variáveis
  - Altera fluxo de execução
- Heap
  - Corrompe a estrutura de controle
  - Sobreescreve o endereço de retorno

Array Bound Check (ABC)

- Técnica de defesa contra BOF
- Garante os limites de arranjo
- Sobrecarga

```
...
int buffer[MAX];
int i,j,a;
...
for(i = 0; i < j; i++)
{
        ...
        buffer[i] = a;
        ...
}
```

```
...
int buffer[MAX];
int i,j,a;
...
for(i = 0; i < j; i++)
{
        ...
        if(i > 0 && i < MAX)
                buffer[i] = a;
        ...
}
```

Figura 2. Código Vulnerável x Código utilizando ABC.

Microprocessador MIPS

- Desenvolvido em *Stanford* (1982~1984)
  - Arquitetura RISC
    - ***Reduced Instruction Set Computer***
    - Tamanho fixo de instrução
  - *pipeline* em nível de hardware
    - Lista de operações criadas por hardware
    - Cria uma lista dos processos no cache do processador

Microprocessador MIPS – 16 bits

- Versão mais simples
- Menor número de instruções
  - 16 bits cada instrução
- Menor número de registradores
  - 16
- Endereços limitados
  - 64KB de dados
- Opcode
  - Instrução para realizar tarefas

MIPS – Instruções

Operações de 16 bits, dos quais, 4 são opcode

Instruções são de 3 tipos:

- Tipo R - Aritméticas
  - 3 registradores para instrução
- Tipo I - Memória
  - 2 registradores para instrução
  - 1 registrador para constante
- Tipo J – Sem registradorse
  - 12 bits paraconstantes

MIPS – Instruções

Para o trabalhado foram implementadas as seguintes instruções:

- Add
- Sub
- Addi
- And
- Or
- Not
- Shiftl
- Shitftr
- Lw
- Sw
- Beq
- Jump
- Halt
- nop

Array Bound Check (ABC)

- Em Assembly – Arquitetura MISP
  - Aumento de operações b*ranch*
  - Cálculo dos endereços limites para comparação

```
LOOP:
    ...
    sw $R1,0($R2)
    ..
    jmp LOOP
```

```
    addi $R3, $R0, MAX;   tamanho do array
    add  $R4, $R2, $R3;   $R2 contém o endereço base do arranjo
LOOP:
    ...
    bgt  $R2, $R4, AFTER
    bgt  $R0, $R2, AFTER
    sw $R1,0($R2)
AFTER:
    ...
    jmp LOOP
```

Trabalhos na Área

- Análise estática
  - Durante desenvolvimento
- Análise dinâmica
  - Utiliza o hardware
  - Shao - arquitetura DLX

Trabalhos na Área

- Shao - arquitetura DLX
  - Instruções especias para limites
  - 3 registradores – limites e local de acesso
  - Se houver violação -> interrupção
- Neste trabalho
  - Safe Store Word
  - Apenas limite superior
  - Opera em apenas uma instrução
  - 3 registradores

Projeto - SSW

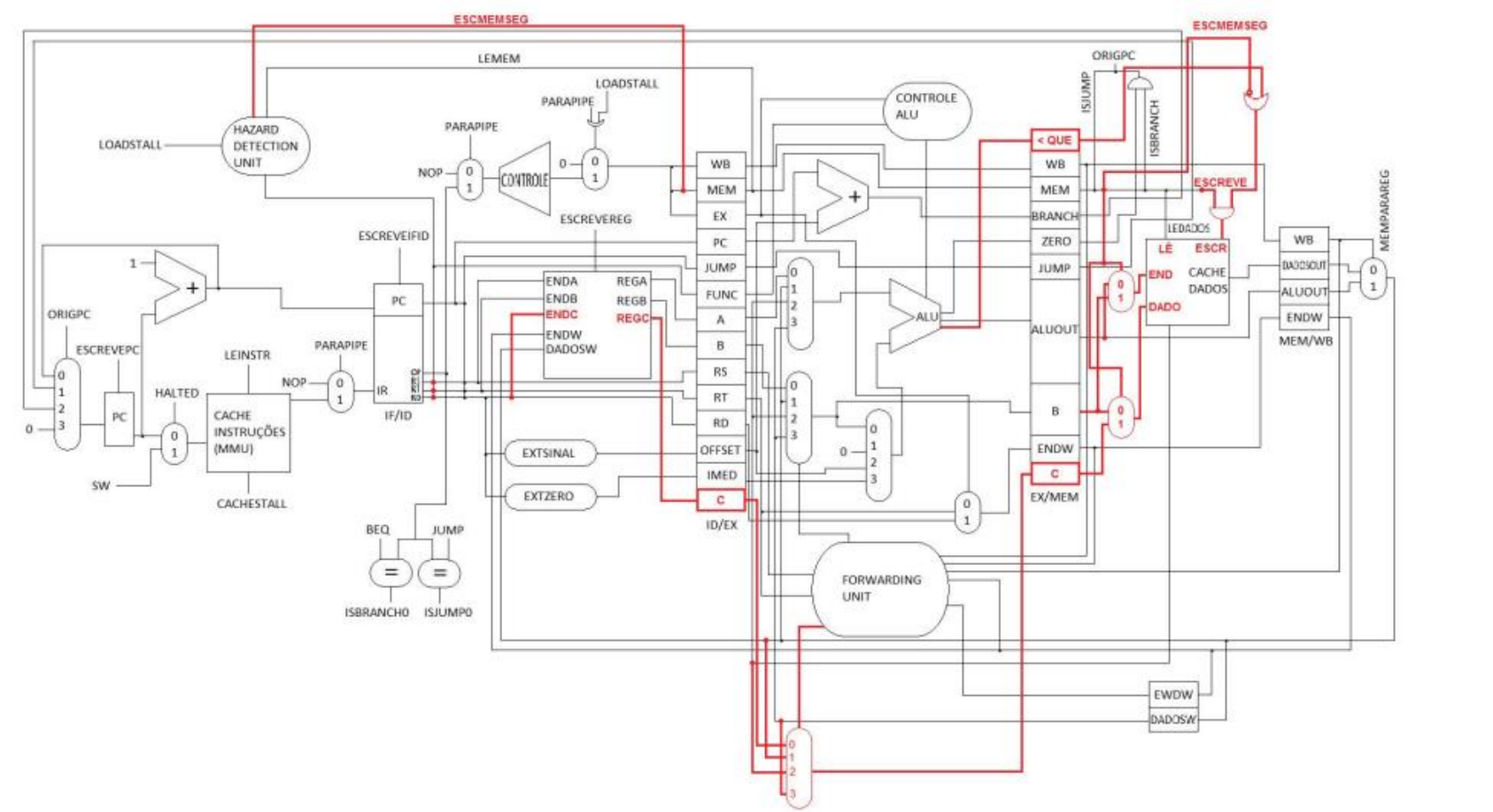
Tabela 1. Estágios de Pipeline.

Estágio do pipeline	Operações
Fetch	$IR = MEM[PC]$ $PC = PC + 1$
Decode	$A = rs$ $B = rt$ $C = rd$
Execute	$A - B$
Memory	$if((A - B) > 0)$ MEM$[B] = C$ $else$ INT
Write-back	-

Projeto – SSW

- Desenvolvimento caminho de dados
  - HDL
  - Cyclone 2 - Altera
  - 1663 linhas
- Inserção da estrutura do trabalho
  - 1710 linhas
  - 2.83% crescimento
  - 2,43% sobrecarga

Projeto – SSW


ESCMEMSEG
LEMEM
LOADSTALL
PARAPIPE
HAZARD DETECTION UNIT
NOP
CONTROLE
ESCREVEREG
ESCREVEIFID
CONTROLE ALU
ORIGPC
ISJUMP
ISBRANCH
< QUE
WB
MEM
BRANCH
ZERO
JUMP
ALUOUT
B
ENDW
C
EX/MEM
ESCREVE
LEDADOS
LÊ
ESCR
END
CACHE DADOS
DADO
MEMPARAREG
DADOSOUT
MEM/WB
ENDA
ENDB
ENDC
ENDW
DADOSW
REGA
REGB
REGC
EX
PC
FUNC
A
RS
RT
RD
OFFSET
IMED
ID/EX
ALU
ESCREVEPC
LEINSTR
HALTED
SW
CACHE INSTRUÇÕES (MMU)
CACHESTALL
IR
IF/ID
EXTSINAL
EXTZERO
BEQ
JUMP
ISBRANCH0
ISJUMP0
FORWARDING UNIT
EWDW
DADOSW

Teste Comparativo

- **Original** - os arranjos são acessados sem ABC
- **Baseline** - é utilizado um ABC em software (verificando apenas o limite superior).
- **SSW** - o programa será modificado de forma que utilize a nova instrução.

arraycopy	Instruções	Sobrecarga(%)
original	282	
baseline	346	22,7
SSW	288	2,1

Tabela 2. Resultados dos testes.

Algoritmo de Teste

- *Copyarray*
  - De um array de 32 bytes para um de 16 bytes
  - Preenchido com a palavra CODE
  - Byte a byte
- Original sem bloqueio
- *Baseline* e SSW bloqueiam – não geram interrupção

Trabalhos Futuros

- Validar limite inferior
- Implementar SLW (Safe Load Word)
- Analisar influência no desempenho

Referências

- TORRES, Eric S. et al. Implementaç ao em Hardware de Instruç ao Segura de Acesso a Memória-Caso MIPS 16 bit.