# DECODIFICADOR RÁPIDO PARA PROCESSADORES MODELADOS EM ARCHC

Orientador: Rodolfo Jardim de Azevedo
Aluno: Rafael Madeira

INSTITUTO DE COMPUTAÇÃO - IC
Sae/Unicamp

## Introdução:

**Arquitetura de Computadores - Processadores - ArchC**

ArchC é uma linguagem para descricão de arquiteturas desenvolvida no laboratório LSC - IC - Unicamp, baseada em SystemC, uma linguagem de descrição de hardware. Este trabalho de iniciacão científica consistiu em realizar alterações no código fonte do ArchC com a finalidade de obtermos ganho de desempenho ao simularmos arquiteturas de processadores descritas em ArchC. Tais alterações resultaram na substituição do atual processo de decodificação dinâmica por um novo processo, estático (mais rápido), para decodificação das instruções. Neste trabalho nos focamos em arquiteturas descritas em ArchC cujas ISAs possuem instruções de tamanho fixo igual à 32 bits. Todo código fonte foi escrito na linguagem C.

## Metodologia:

Figura 1: Campos de instrução do Tipo I do processador MIPS

| opcode | rs | rt | address |
|---|---|---|---|
| 6 | 5 | 5 | 16 |

Passo 1: Encontrar nas ISAs dos processadores os formatos de instruções dos processadores, com seus respectivos campos.

Figura 2: Estruturas para realização da decodificação estática de instruções, para o modelo MIPS

```
1  typedef struct{
2    //Type I instructions fields
3    unsigned imm:16;
4    unsigned rt:5;
5    unsigned rs:5;
6    unsigned op:6;
7  }T_Type_I;
8
9  typedef struct{
10   //Type R instructions fields
11   unsigned func:6;
12   unsigned shamt:5;
13   unsigned rd:5;
14   unsigned rt:5;
15   unsigned rs:5;
16   unsigned op:6;
17 }T_Type_R;
18
19 typedef struct{
20   //Type J instructions fields
21         .
22         .
23         .
24
25 }T_Type_J;
26
27 typedef union mips1_types{
28   T_Type_R Type_R;
29   T_Type_I Type_I;
30   T_Type_J Type_J;
31   unsigned int instr;
32 }mips1_Buffer;
```

Passo 2: Construir, automaticamente, estruturas para cada formato de instrução, com o objetivo de ncapsulá-las em um única estrutura (linhas 37 à 32).

Figura 3: Processo de decodificação do modelo para processador MIPS

```
1  mips1_Buffer mips1_instr;
2  mips1_instr.instr = decoder->prog_source-> GetBits(buffer,&quant
       ,31,32,0);
3
4  switch(mips1_instr.Type_R.op){
5
6  case 36:
7    //instruction lbu:
8    type = 1;
9    instruction = &(mips1_parms::mips1_isa::instructions[0]);
10   break;
11 case 32:
12   //instruction lb:
13   type = 1;
14   instruction = &(mips1_parms::mips1_isa::instructions[1]);
15   break;
16
17        .
18        .
19        .
20
21
22 case 0:
23   switch(mips1_instr.Type_R.func){
24
25   case 32:
26     //instruction add:
27     type = 0;
28     instruction = &(mips1_parms::mips1_isa::instructions[20]);
29     break;
30   case 33:
31     //instruction addu:
32     type = 0;
33     instruction = &(mips1_parms::mips1_isa::instructions[21]);
34     break;
35   }//end switch for 'func' field
36 case 2:
37   //instruction j:
38   type = 2;
39   instruction = &(mips1_parms::mips1_isa::instructions[45]);
40   break;
41        .
42        .
43        .
44 }//end main switch
45        .
46        .
47        .
48
49 }//end FindAndDecodeAs
```

Passo 3: Utilizando as estruturas criadas no passo 2, criamos, automaticamente, uma forma de decodificação estática com estruturas *switch-case*.

Figura 4: Processo de decomposição dos bits de uma instrução, para o processador MIPS.

```
1  switch(type){
2  case 0: //Type_R
3    .
4    .
5    .
6    break;
7  case 1: //Type_I
8    //field op
9    fields[1] = mips1_instr.Type_R.op;
10   //field rs
11   fields[2] = mips1_instr.Type_R.rs;
12   //field rt
13   fields[3] = mips1_instr.Type_R.rt;
14   //field imm
15   i = mips1_instr.Type_I.imm;
16   mask = 1 << 15;
17   if (i & mask){
18     mask = 0xffffffff;
19     mask = mask << 16;
20     i = i | mask;
21   }//if
22   fields[7] = i;
23   break;
24 case 2: //Type_J
25   .
26   .
27   .
28   break;
29 }//switch;
```

Passo 4: Decomposição da instrução ( já decodificada ) nos bits referentes aos campos do seu formato.

## Resultados:

Nas tabelas 1.1 e 1.2 mostramos, utilizando decodificação estática e dinâmica respectivamente,o ganho de desempenho total, ou seja, quando todo o processo de decodificacão de instrucões é migrado para a forma estática. Para coleta dos dados foram utilizados 6 benchmarks do pacote Mibenchdo processador MIPS.Apresentamos a média, mediana, desvio padrão e o ganho em porcentagemem função dos valores da mediana de um grande conjunto de dados onde cada umrepresentava o número de ciclos gastos pelo processador durante o processo.

Decodificador Estático

| | Média | Mediana | Desv. Padrão |
|---|---|---|---|
| automotive/qsort | 4202,79 | 3632 | 2754,45 |
| consumer/jpeg | 7342,78 | 3928 | 30962,15 |
| network/dijkstra | 4307,58 | 3652 | 6066,15 |
| security/rijndael | 5218,73 | 3912 | 53132,42 |
| office/stringsearch | 5731,11 | 3644 | 36621,17 |
| telecomm/gsm | 4180,91 | 3624 | 5131,34 |
| MÉDIA | **4191,85** | **3628** | **3942,9** |

*Tabela 1.1: Desempenho do processo completo de decodificação estática em números de ciclos gastos pelo processador, em um modelo MIPS.*

Decodificador Dinâmico

| | Média | Mediana | Desv. Padrão | % Mediana |
|---|---|---|---|---|
| automotive/qsort | 8317,78 | 6488 | 5843,96 | **178,63** |
| consumer/jpeg | 15004,98 | 7104 | 220406,83 | **180,86** |
| network/dijkstra | 8857,87 | 6900 | 14759,29 | **188,94** |
| security/rijndael | 9766,47 | 8628 | 10230,99 | **220,55** |
| office/stringsearch | 9054,2 | 6448 | 39114,53 | **176,95** |
| telecomm/gsm | 9325,25 | 6826 | 51140,29 | **188,36** |
| MÉDIA | **8821,52** | **6657** | **28492,13** | **183,49** |

*Tabela 1.2: Desempenho do processo completo de decodificação dinâmica em números de ciclos gastos pelo processador, em um modelo MIPS.*

## Problemas - ARM:

O processador ARM pertence à família RISC e possui um modelo escrito em ArchC. Entretanto, sua ISA estabelece uma forma de decodificação para suas instruções extremamente complexo. Nossos estudos indicam que para a realização da decodificação estática das instruções do Arm são necessárias alterações mais profundas no código fonte do ArchC.

## Conclusão:

De acordo com os resultados apresentados, verificou-se que o ganho obtido ao migrarmos o processo de decodificação para a forma estática forneceu ganhos significativos de desempenho. Apeser de mostrarmos somente dados para o modelo do processador MIPS, testes mostraram que obtemos execução correta e ganhos significativos de desempenho também para os modelos SPARC e PowerPC.