Organização e Arquitetura de Computadores
Trabalho III: Simulador MIPS

Aluno: Gabriel Martins de Miranda
Matricula: 13/0111350
Turma: C

1. Objetivos

→ implementar um simulador da arquitetura MIPS em linguagem de alto nível (C/C++/Java).

→ implementar as funções de busca da instrução (fetch()), decodificação da instrução (decode()) e execução da instrução (execute()).

→ ler arquivos binários contendo o segmento de código e o segmento de dados para a memória definida no programa e executá-lo.

→ compilador: gcc (Ubuntu 4.8.4-2ubuntu1~14.04) 4.8.4

2. Funções

- FUNÇÕES NOVAS

→ *void fetch();*

//lê uma instrução da memória e coloca-a em ri, atualizando o pc para apontar para a próxima instrução (soma 4).

→ *void decode();*

//extrair todos os campos da instrução:

opcode //operacao basica da instrucao: opcode
rs //primeiro registrador de operando origem
rt //segundo registrador de operando origem
rd //registrador de operando destino: resultado
shamt //deslocamento: shift amount
funct //variacao da operacao: function code
k16 //constante de 16 bits, valor imediato em instruções tipo I

k26   //constante de 26 bits, para instruções tipo J

→ void execute();
//executa a instrução lida por fetch() e decodificada por decode(). As possíveis instruções são as seguintes:

```
LH=0x21   ,         // Load Halfword
SB=0x28   ,         // Store Byte
BLEZ=0x06 ,         // Branch on Less Than or Equal to Zero
SLTIU=0x0B,         // Set on Less Than Immediate Unsigned
J=0x02    ,         // Jump
JAL=0x03  ,         // Jump and Link
LW=0x23   ,         // Load Word
LB=0x20   ,         // Load Byte
LBU=0x24  ,         // Load Byte Unsigned
LHU=0x25  ,         // Load Halfword Unsigned
LUI=0x0F  ,         // Load Upper Immediate
SW=0x2B   ,         // Store Word
SH=0x29   ,         // Store HalfWord
BEQ=0x04  ,         // Branch on Equal
BNE=0x05  ,         // Branch on Not Equal
BGTZ=0x07 ,         // Branch on Greater Than Zero
ADDI=0x08 ,         // Add Immediate Word
ADDIU=0x09,         // Add Immediate Unsigned Word
SLTI=0x0A ,         // Set on Less Than Immediate
ANDI=0x0C ,         // And Immediate
ORI=0x0D  ,         // Or Immediate
XORI=0x0E ,         // Exclusive OR Immediate

//caso em que o opcode é 0 (EXT=0x00  ,         // Geral)
ADD=0x20    ,       // Add Word
SUB=0x22    ,       // Subtract Word
MULT=0x18   ,       // Multiply Word
DIV=0x1A    ,       // Divide Word
AND=0x24    ,       // And
MFLO=0x12   ,       // Move From LO Register
OR=0x25     ,       // Or
XOR=0x26    ,       // Exclusive OR
NOR=0x27    ,       // Not Or
```

SLT=0x2A , *// Set on Less Than*
JR=0x08 , *// Jump Register*
SLL=0x00 , *// Shift Word Left Logical*
SRL=0x02 , *// Shift Word Right Logical*
SRA=0x03 , *// Shift Word Right Arithmetic*
SYSCALL=0x0c, *// System Call*
MFHI=0x10 , *// Move From HI Registercont_data*

→ *void step();*
//executa uma instrução do MIPS: step() => fecth(), decode(), execute()

→ *void run();*
//executa o programa até encontrar uma chamada de sistema para encerramento, ou até o pc ultrapassar o limite do segmento de código (4k words).

→ *void dump_mem(int start, int end, char format);*
//Imprime o conteúdo da memória a partir do endereço start até o endereço end. O formato pode ser em hexa ('h') ou decimal ('d').

→ *void dump_reg(char format);*
//Imprime o conteúdo dos registradores do MIPS, incluindo o banco de registradores e os registradores pc, hi e lo. O formato pode ser em hexa ('h') ou decimal ('d').

● FUNÇÕES ANTIGAS

- Foram utilizadas as funções do trabalho anterior, porém elas foram modificadas para se adequar ao simulador. Compõem as funções de acesso à memória.

→ int32_t lb(uint32_t *address, uint16_t kte);
→ int32_t lh(uint32_t *address, uint16_t kte);
→ int32_t lw(uint32_t *address, uint16_t kte);
→ void sb(uint32_t *address, uint16_t kte, int8_t dado);
→ void sh(uint32_t *address, uint16_t kte, int16_t dado);
→ void sw(uint32_t *address, uint16_t kte, int32_t dado);

3. Testes e resultados

Dentre os programas testados, os seguintes resultados foram encontrados:

1 - primos.asm – programa que imprime os oitos primeiros números primos. Os resultados foram satisfatórios e corresponderam aos valores mostrados no MARS.

```
Os oito primeiros numeros primos sao :
0x00000001

0x00000003

0x00000005

0x00000007

0x0000000b

0x0000000d

0x00000011

0x00000013


fim
```

```
mem[2048] = 00000001
mem[2049] = 00000003
mem[2050] = 00000005
mem[2051] = 00000007
mem[2052] = 0000000b
mem[2053] = 0000000d
mem[2054] = 00000011
mem[2055] = 00000013
mem[2056] = 00000008
mem[2057] = 6f20734f
mem[2058] = 206f7469
mem[2059] = 6d697270
mem[2060] = 6f726965
mem[2061] = 756e2073
mem[2062] = 6f72656d
mem[2063] = 72702073
mem[2064] = 736f6d69
mem[2065] = 6f617320
mem[2066] = 00203a20
mem[2067] = 00000020
mem[2068] = 00000000
mem[2069] = 00000000
mem[2070] = 00000000
mem[2071] = 00000000
mem[2072] = 00000000
mem[2073] = 00000000
mem[2074] = 00000000
mem[2075] = 00000000
mem[2076] = 00000000
mem[2077] = 00000000
mem[2078] = 00000000
mem[2079] = 00000000
mem[2080] = 00000000
```

```
breg[0]  = 00000000
breg[1]  = 00000000
breg[2]  = 0000000a
breg[3]  = 00000000
breg[4]  = 0000204c
breg[5]  = 00000000
breg[6]  = 00000000
breg[7]  = 00000000
breg[8]  = 00002020
breg[9]  = 00000000
breg[10] = 00000000
breg[11] = 00000000
breg[12] = 00000000
breg[13] = 00000000
breg[14] = 00000000
breg[15] = 00000000
breg[16] = 00000000
breg[17] = 00000000
breg[18] = 00000000
breg[19] = 00000000
breg[20] = 00000000
breg[21] = 00000000
breg[22] = 00000000
breg[23] = 00000000
breg[24] = 00000000
breg[25] = 00000000
breg[26] = 00000000
breg[27] = 00000000
breg[28] = 00001800
breg[29] = 00003ffc
breg[30] = 00000000
breg[31] = 00000000
pc       = 00000048
hi       = 00000000
lo       = 00000000
```

```
Os oito primeiros numeros primos sao : 1 3 5 7 11 13 17 19
-- program is finished running --
```

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00002000	0x00000001	0x00000003	0x00000005	0x00000007	0x0000000b	0x0000000d	0x00000011	0x00000013
0x00002020	0x00000008	0x6f20734f	0x206f7469	0x6d697270	0x6f726965	0x756e2073	0x6f72656d	0x72702073
0x00002040	0x736f6d69	0x6f617320	0x00203a20	0x00000020	0x00000000	0x00000000	0x00000000	0x00000000
0x00002060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000021a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000021c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000021e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000204c
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00002020
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00003ffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00000048
hi		0x00000000
lo		0x00000000

2 - fibonacci.asm - imprime alguns números da série de Fibonacci.
Tudo saiu como esperado.

```
1 - step()
2 - run()
3 - sair
2

0x00000001

0x00000001

0x00000002

0x00000003

0x00000005

0x00000008

0x0000000d

0x00000015

0x00000022

0x00000037

0x00000059

0x00000090


fim
```

```
mem[2048] = 00000001
mem[2049] = 00000001
mem[2050] = 00000002
mem[2051] = 00000003
mem[2052] = 00000005
mem[2053] = 00000008
mem[2054] = 0000000d
mem[2055] = 00000015
mem[2056] = 00000022
mem[2057] = 00000037
mem[2058] = 00000059
mem[2059] = 00000090
mem[2060] = 0000000c
mem[2061] = 68540020
mem[2062] = 69462065
mem[2063] = 616e6f62
mem[2064] = 20696363
mem[2065] = 626d756e
mem[2066] = 20737265
mem[2067] = 3a657261
mem[2068] = 0000000a
mem[2069] = 00000000
mem[2070] = 00000000
mem[2071] = 00000000
mem[2072] = 00000000
mem[2073] = 00000000
mem[2074] = 00000000
mem[2075] = 00000000
mem[2076] = 00000000
mem[2077] = 00000000
mem[2078] = 00000000
mem[2079] = 00000000
mem[2080] = 00000000
```

```
breg[0] = 00000000
breg[1] = 00000000
breg[2] = 0000000a
breg[3] = 00000000
breg[4] = 00002034
breg[5] = 0000000c
breg[6] = 00000000
breg[7] = 00000000
breg[8] = 00002030
breg[9] = 00000000
breg[10] = 00000090
breg[11] = 00000037
breg[12] = 00000059
breg[13] = 0000000c
breg[14] = 00000000
breg[15] = 00000000
breg[16] = 00000000
breg[17] = 00000000
breg[18] = 00000000
breg[19] = 00000000
breg[20] = 00000000
breg[21] = 00000000
breg[22] = 00000000
breg[23] = 00000000
breg[24] = 00000000
breg[25] = 00000000
breg[26] = 00000000
breg[27] = 00000000
breg[28] = 00001800
breg[29] = 00003ffc
breg[30] = 00000000
breg[31] = 00000044
pc       = 0000004c
hi       = 00000000
lo       = 00000000
```

```
The Fibonacci numbers are:
1 1 2 3 5 8 13 21 34 55 89 144
-- program is finished running --
```

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00002000	0x00000001	0x00000001	0x00000002	0x00000003	0x00000005	0x00000008	0x0000000d	0x00000015
0x00002020	0x00000022	0x00000037	0x00000059	0x00000090	0x0000000c	0x68540020	0x69462065	0x616e6f62
0x00002040	0x20696363	0x626d756e	0x20737265	0x3a657261	0x0000000a	0x00000000	0x00000000	0x00000000
0x00002060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000021a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000021c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000021e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x00002000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00002034
$a1	5	0x0000000c
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00002030
$t1	9	0x00000000
$t2	10	0x00000090
$t3	11	0x00000037
$t4	12	0x00000059
$t5	13	0x0000000c
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00003ffc
$fp	30	0x00000000
$ra	31	0x00000044
pc		0x0000004c
hi		0x00000000
lo		0x00000000

3 - testando.asm - programa que testa todas as instruções implementadas. Aqui novamente os resultados foram satisfatórios, como se pode ver nas imagens.

```
Os tres primeiros numeros pares sao :
0x00140000
:
0x00000002
:
0x00000004
:
0xfffffff1
0x00000000
0x00000064
0x00000000
0x00000004
0x00000004
0x00000015
0x00000011
0xffffffea
0x0000012c
0x00000004
0x00000000
0x00000000
0x00000000
0x00000000
0x00140000
0x00140000
0x00140000
0x00140000

fim
```

```
mem[2048] = 00140000
mem[2049] = 00000002
mem[2050] = 00000004
mem[2051] = 00000003
mem[2052] = 7420734f
mem[2053] = 20736572
mem[2054] = 6d697270
mem[2055] = 6f726965
mem[2056] = 756e2073
mem[2057] = 6f72656d
mem[2058] = 61702073
mem[2059] = 20736572
mem[2060] = 206f6173
mem[2061] = 2000203a
mem[2062] = 00000000
mem[2063] = 00000000
mem[2064] = 00000000
mem[2065] = 00000000
mem[2066] = 00000000
mem[2067] = 00000000
mem[2068] = 00000000
mem[2069] = 00000000
mem[2070] = 00000000
mem[2071] = 00000000
mem[2072] = 00000000
mem[2073] = 00000000
mem[2074] = 00000000
mem[2075] = 00000000
mem[2076] = 00000000
mem[2077] = 00000000
mem[2078] = 00000000
mem[2079] = 00000000
mem[2080] = 00000000
```

```
breg[0]  = 00000000
breg[1]  = 00000000
breg[2]  = 0000000a
breg[3]  = 00000001
breg[4]  = 00140000
breg[5]  = 00000000
breg[6]  = 00000000
breg[7]  = 00000000
breg[8]  = 00002000
breg[9]  = 00000061
breg[10] = 00000005
breg[11] = 00000000
breg[12] = 00000000
breg[13] = 00000000
breg[14] = 00000000
breg[15] = 00000000
breg[16] = 00000000
breg[17] = 00000000
breg[18] = 00000000
breg[19] = 00000000
breg[20] = 00000000
breg[21] = 00000000
breg[22] = 00000000
breg[23] = 00000000
breg[24] = 00000000
breg[25] = 00000000
breg[26] = 00000000
breg[27] = 00000000
breg[28] = 00001800
breg[29] = 00003ffc
breg[30] = 00000000
breg[31] = 00000120
pc       = 00000160
hi       = 00000000
lo       = 00000004
```

```
Os tres primeiros numeros pares sao : 0 2 4 -1501000442117-22300400001310720131072013107201310720
-- program is finished running --
```

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00002000	0x00140000	0x00000002	0x00000004	0x00000003	0x7420734f	0x20736572	0x6d697270	0x6f726965
0x00002020	0x756e2073	0x6f72656d	0x61702073	0x20736572	0x206f6173	0x2000203a	0x00000000	0x00000000
0x00002040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000021a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000021c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000021e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000001
$a0	4	0x00140000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00002000
$t1	9	0x00000061
$t2	10	0x00000005
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x00001800
$sp	29	0x00003ffc
$fp	30	0x00000000
$ra	31	0x00000120
pc		0x00000160
hi		0x00000000
lo		0x00000004