Home work 3 - disassembly

GPR\_file and ALU

Student: Otabek Sherman

ID: 326868775

Here is the MIPS binary code you need to dis-assembly:

IMem adrs => IMem data

x"00400000" => x"20010001"

x"00400004" => x"20020002"

x"00400008" => x"20030003"

x"0040000C" => x"20040004"

x"00400010" => x"20050005"

x"00400014" => x"20060006"

x"00400018" => x"20070007"

x"0040001C" => x"20080008"

x"00400020" => x"20090009"

x"00400024" => x"200A000A"

x"00400028" => x"200B000B"

x"0040002C" => x"200C000C"

x"00400030" => x"200D000D"

x"00400034" => x"200E000E"

x"00400038" => x"200F000F"

x"0040003C" => x"20100010"

x"00400040" => x"20110001"

x"00400044" => x"20120002"

x"00400048" => x"20130004"

x"0040004C" => x"20140008"

x"00400050" => x"20150010"

x"00400054" => x"20160020"

x"00400058" => x"20170040"

x"0040005C" => x"20180080"

x"00400060" => x"20190100"

x"00400064" => x"201B0200"

x"00400068" => x"201C0400"

x"0040006C" => x"201D0800"

x"00400070" => x"201E1000"

x"00400074" => x"201F2000"

x"00400078" => x"02018020"

x"0040007C" => x"02028020"

x"00400080" => x"02038020"

x"00400084" => x"02048020"

x"00400088" => x"00000000"

x"0040008C" => x"00000000"

x"00400090" => x"00000000"

x"00400094" => x"00220820"

x"00400098" => x"00000000"

x"0040009C" => x"00000000"

x"004000A0" => x"00230820"

x"004000A4" => x"00000000"

x"004000A8" => x"00000000"

x"004000AC" => x"00810820"

x"004000B0" => x"00000000"

x"004000B4" => x"00000000"

x"004000B8" => x"00A10820"

x"004000BC" => x"00000000"

x"004000C0" => x"00000000"

x"004000C4" => x"00260820"

x"004000C8" => x"00000000"

x"004000CC" => x"00000000"

x"004000D0" => x"00E83820"

x"004000D4" => x"00000000"

x"004000D8" => x"00000000"

x"004000DC" => x"00E90820"

x"004000E0" => x"00000000"

x"004000E4" => x"00000000"

x"004000E8" => x"002A0820"

x"004000EC" => x"00000000"

x"004000F0" => x"00000000"

x"004000F4" => x"002B0820"

x"004000F8" => x"00000000"

x"004000FC" => x"00000000"

x"00400100" => x"002C0820"

x"00400104" => x"00000000"

x"00400108" => x"00000000"

x"0040010C" => x"002D0820"

x"00400110" => x"00000000"

x"00400114" => x"00000000"

x"00400118" => x"002E0820"

x"0040011C" => x"00000000"

x"00400120" => x"00000000"

x"00400124" => x"002F0820"

x"00400128" => x"00000000"

x"0040012C" => x"00000000"

x"00400130" => x"00008020"

x"00400134" => x"200F0003"

x"00400138" => x"00000000"

x"0040013C" => x"00000000"

x"00400140" => x"00000000"

x"00400144" => x"22100001"

x"00400148" = > x"00000000"

x"0040014C" => x"21EFFFFF"

x"00400150” = > x”11E00004"

x"00400154” = > x”00000000"

x"00400158” = > x”00000000"

x"0040015C” = > x”08100051"

x"00400160” = > x”00000000"

x"00400164” = > x”08100059"

x"00400168” = > x”00000000"

x"0040016C” = > x”00000000"

See questions below:

1. What does the code in addresses 0x400000-0x40003C do?

Code in addresses 0x400000-0x40003C contains memory addresses of registers 1-16. The commands in these addresses writes the values into the registers.

To reg 1 writes 1

To reg 2 writes 2

…

To reg 16 writes 16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IMem adrs | op | rs | rt | rd |
| x"00400000" | addi | R1 | $zero | 1 |
| x"00400004" | addi | R2 | $zero | 2 |
| x"00400008" | addi | R3 | $zero | 3 |
| x"0040000C" | addi | R4 | $zero | 4 |
| x"00400010" | addi | R5 | $zero | 5 |
| x"00400014" | addi | R6 | $zero | 6 |
| x"00400018" | addi | R7 | $zero | 7 |
| x"0040001C" | addi | R8 | $zero | 8 |
| x"00400020" | addi | R9 | $zero | 9 |
| x"00400024" | addi | R10 | $zero | 10 |
| x"00400028" | addi | R11 | $zero | 11 |
| x"0040002C" | addi | R12 | $zero | 12 |
| x"00400030" | addi | R13 | $zero | 13 |
| x"00400034" | addi | R14 | $zero | 14 |
| x"00400038" | addi | R15 | $zero | 15 |
| x"0040003C" | addi | R16 | $zero | 16 |

1. What does the code in addresses 0x400040-0x400074 do?

The addresses 0x400040-0x400074 contain the memory addresses of registers 7-31. The commands in these addresses writes the values into the registers the powers of 2: {1, 2, 4, …, 8192}.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IMem adrs | op | rd | rs | rt |
| x"00400040" | addi | R1 | $zero | 1 |
| x"00400044" | addi | R2 | $zero | 2 |
| x"00400048" | addi | R3 | $zero | 4 |
| x"0040004C" | addi | R4 | $zero | 8 |
| x"00400050" | addi | R5 | $zero | 16 |
| x"00400054" | addi | R6 | $zero | 32 |
| x"00400058" | addi | R7 | $zero | 64 |
| x"0040005C" | addi | R8 | $zero | 128 |
| x"00400060" | addi | R9 | $zero | 256 |
| x"00400064" | addi | R10 | $zero | 512 |
| x"00400068" | addi | R11 | $zero | 1024 |
| x"0040006C" | addi | R12 | $zero | 2048 |
| x"00400070" | addi | R13 | $zero | 4096 |
| x"00400074" | addi | R14 | $zero | 8192 |

1. What does the code in addresses 0x400078-0x400090 do and what will be the contents of register $16 when we reach address 0x400094?

The addresses 0x400078-0x400084 contains the add operations where we add the values of registers R1-R4 to register R16 and save result in register 16:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IMem adrs | op | rd | rs | rt |
| x"00400078" | add | R16 | R16 | R1 |
| x"0040007c" | add | R16 | R16 | R2 |
| x"00400080" | add | R16 | R16 | R3 |
| x"00400084" | add | R16 | R16 | R4 |

We program doesn’t do nothing in addresses 0x400088-0x400090.

In the final address the program adds R1 to R2 and writes the result in R1

After calculation the final result in register $16 will be:

1+2+3+4+16=26

1. What does the code in addresses 0x400144-0x400168 do and what will be the contents of register $16 at the end of this code section?

After the running of the previous code the register $15 has the value 18 and the register %16 has the value 26.

The code in the addresses 0x400144-0x400168 implements a loop.

The loop running while the value of register R15 is more than 0.

In every iteration we adds 1 to the value of register R16 and adds -1 to the value of the register R15.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IMem adrs | op | rs | rt | imm |
| 0x400144 | addi | R16 | R16 | 1 |
| 0x400148 | nop |  |  |  |
| 0x40014c | addi | R15 | R15 | -1 |
| 0x400150 | beq | R15 | 0x400164 |  |
| 0x400154 | nop |  |  |  |
| 0x400158 | nop |  |  |  |
| IMem adrs | op | imm | | |
| 0x40015c | j | 0x400144 | | |
| 0x400160 | nop |  |  |  |
| 0x400164 | j | 0x400164 | | |
| 0x400168 | nop |  |  |  |

1. Does this code tests the GPR\_file and ALU parts of a MIPS CPU? How? What is not covered?

This code tests the GPR\_file part and ALU parts of a MIPS CPU.

When we reading the values from the registers the code tests the GPU\_file. The commands add, addi, beqz are testing the ALU part.