**Homework 3 Answer Sheet for Q1 – Q4**

Please state the name and SID of all members of your group (you can add more rows if there are more than 5 members).

|  |  |  |
| --- | --- | --- |
| member | name | SID |
| #1 (contact person) | WU Xiaoqing | 56641363 |
| #2 | HUANG Jinyu | 56641166 |
| #3 | WU Jianrui | 56641885 |
| #4 | WANG Chenchen | 56344983 |
| #5 | DU Junye | 56641800 |

**Q1:**

1. Please explain how many types of instructions are supported in your processor, and explain the format of each type of instructions (e.g., which bits are used as the operation & function code, which bits are used to index the 1st, 2nd or 3rd operand, and which bits are used to store the immediate number).

Our processor supports 3 types of instructions, which are R-type, I-type and J-type.

Each type of instruction has a 4-bit OP code, represented by bit 15-12. Specially, the OP code of R-type is 0000 constantly. R-type instructions use bit 11-9 as rs (1st  operand), bit 8-6 as rt (2nd operand), bit 5-3 as rd (3rd operand), and bit 2-0 as function code.

For I-type instructions, they use bit 11-9 as rs (1st  operand), bit 8-6 as rt (2nd operand), and bit 5-0 as address offset or immediate number.

For J-type instructions, they use bit 11-0 as address.

1. Please explain the format of each of the following instructions (including the format of this instruction and its operation & function code).

* li

li $r1 x

function: load immediate number to one register ($r1)

OP code: 1101

Format of instruction: 1101 000 $r1 x

* add

add $r1, $r2, $r3

function: add two register numbers ($r2 and $r3) and store the result in one register ($r1)

OP code: 0000

function code: 000

Format of instruction: 0000 $r2 $r3 $r1 000

* neg

neg $r1, $r2

function: negate a register number ($r2) and store the result in the register ($r1)

OP code: 0000

function code: 111

Format of instruction: 0000 $r2 000 $r1 111

* ble

ble $r1, $r2, x

function: if the first register number ($r1) is smaller than the second register number ($r2), then jump to address (PC + x)

OP code: 1100

Format of instruction: 1100 $r1 $r2 x

* halt

halt

function: all registers (including PC, the 8 general purpose registers and all other registers) in the processor are disabled (so the processor halts)

OP code: 1110

Format of instruction: 1110 0000 0000 0000

1. Please fill the following table with the binary machine code of each instruction of the testing program:

|  |  |  |  |
| --- | --- | --- | --- |
| label | instruction | machine code (binary) | machine code (hex ) |
| .text | li $r1, 1 | 1101 0000 0000 0001 | d001 |
|  | li $r2, 10 | 1101 0000 0100 1010 | d04a |
|  | li $r3, 2 | 1101 0000 1000 0010 | d082 |
|  | neg $r1, $r1 | 0000 0000 0000 0111 | 0007 |
|  | add $r2, $r1, $r2 | 0000 0000 0100 1000 | 0048 |
|  | ble $r3, $r2, -1 | 1100 0100 0111 1111 | c47f |
|  | halt | 1110 0000 0000 0000 | e000 |

1. Please answer what is the value stored in each register after executing the program

|  |  |
| --- | --- |
| register | value |
| $r1 | -1 |
| $r2 | 9 |
| $r3 | 2 |
| $r4 | 0 |
| $r5 | 0 |
| $r6 | 0 |
| $r7 | 0 |
| $r8 | 0 |

**Q2:**

1. Please explain the format of each of the following instructions (including the format of this instruction and its operation & function code).

* and

and $r1, $r2, $r3

function: bit-wise logical AND of two register numbers ($r2 and $r3), and store the result in a register ($r1)

OP code: 0000

function code: 001

Format of instruction: 0000 $r2 $r3 $r1 001

* or

or $r1, $r2, $r3

function: bit-wise logical OR of two register numbers ($r2 and $r3), and store the result in a register ($r1)

OP code: 0000

function code: 010

Format of instruction: 0000 $r2 $r3 $r1 010

* mult

mult $r1, $r2, $r3

function: multiply two register numbers ($r2 and $r3), and store the lower 16-bits in a register ($r1)

OP code: 0000

Function code: 011

Format of instruction: 0000 $r2 $r3 $r1 011

* load

ld $r1, $r2

function: load a 16-bit number from data memory to a register ($r1). The data memory address is the value in a register ($r2)

OP code: 1000

Format of instruction: 1000 $r2 $r1 000000

* store

st $r1, $r2

function: load a 16-bit number from a register ($r1) to data memory. The data memory address is the value in a register ($r2)

OP code: 1001

Format of instruction: 1001 $r2 $r1 000000

* move

move $r1, $r2

function: copy the value of one register ($r2) to another register ($r1)

OP code: 1111

Format of instruction: 1111 $r2 $r1 000000

1. Please fill the following table with the binary machine code of each instruction of the testing program:

|  |  |  |  |
| --- | --- | --- | --- |
| label | instruction | machine code (binary) | machine cod (hex) |
| .text | li $r1, 6 | 1101 0000 0000 0110 | d006 |
|  | li $r2, 5 | 1101 0000 0100 0101 | d045 |
|  | and $r3, $r1, $r2 | 0000 0000 0101 0001 | 0051 |
|  | li $r8, 0 | 1101 0001 1100 0000 | d1c0 |
|  | store $r3, $r8 | 1001 0101 1100 0000 | 95c0 |
|  | or $r4, $r1, $r2 | 0000 0000 0101 1010 | 005a |
|  | li $r8, 1 | 1101 0001 1100 0001 | d1c1 |
|  | store $r4, $r8 | 1001 0111 1100 0000 | 97c0 |
|  | mult $r5, $r3, $r4 | 0000 0100 1110 0011 | 04e3 |
|  | li $r8, 2 | 1101 0001 1100 0010 | d1c2 |
|  | store $r5, $r8 | 1001 1001 1100 0000 | 99c0 |
|  | move $r6, $r5 | 1111 1001 0100 0000 | f940 |
|  | li $r8, 1 | 1101 0001 1100 0001 | d1c1 |
|  | load $r7, $r8 | 1000 1101 1100 0000 | 8dc0 |
|  | add $r8, $r6, $r7 | 0000 1011 1011 1000 | 0bb8 |
|  | halt | 1110 0000 0000 0000 | e000 |

1. Please answer what is the value stored in each register after executing the program

|  |  |
| --- | --- |
| register | value |
| $r1 | 6 |
| $r2 | 5 |
| $r3 | 4 |
| $r4 | 7 |
| $r5 | 28 |
| $r6 | 28 |
| $r7 | 7 |
| $r8 | 35 |

**Q3:**

1. Please explain the format of each of the following instructions (including the format of this instruction and its operation & function code).

* addi

addi $r1, $r2, x

function: add a register number ($r2) and an immediate number (x), and store the result in one register ($r1)

OP code: 0001

Format of instruction: 0001 $r2 $r1 x

* andi

andi $r1, $r2, x

function: bit-wise logical AND of a register number ($r2) and an immediate number x, and store the result in a register ($r1)

OP code: 0010

Format of instruction: 0010 $r2 $r1 x

* ori

ori $r1, $r2, x

function: bit-wise logical OR of a register number ($r2) and an immediate number x, and store the result in a register ($r1)

OP code: 0011

Format of instruction: 0011 $r2 $r1 x

* lsl

lsl $r1, $r2, $r3

function: logical-shift a register number ($r2) to left for several digits ($r3), and store the result in a register ($r1)

OP code: 0000

Function code: 100

Format of instruction: 0000 $r2 $r3 $r1 100

* lsr

lsr $r1, $r2, $r3

function: ogical-shift a register number ($r2) to right for several digits ($r3), and store the result in a register ($r1). The shift-in high bits are all 0

OP code: 0000

Function code: 101

Format of instruction: 0000 $r2 $r3 $r1 101

* asr

asr $r1, $r2, $r3

function: arithmetic-shift a register number ($r2) to right for several digits ($r3), and store the result in one register ($r1). The shift-in high bits are the same as the highest bit of the original register number ($r2)

OP code: 0000

Function code: 110

Format of instruction: 0000 $r2 $r3 $r1 110

* max

max $r1, $r2, $r3

function: get the maximum between two register numbers ($r2, $r3) and store the result in a register ($r1).

OP code: 0101

Format of instruction: 0101 $r2 $r3 $r1 000

* min

min $r1, $r2, $r3

function: get the minimum between two register numbers ($r2, $r3) and store the result in a register ($r1)

OP code: 0110

Format of instruction: 0110 $2 $3 $1 000

1. Please fill the following table with the binary machine code of each instruction of the testing program:

|  |  |  |  |
| --- | --- | --- | --- |
| label | instruction | machine code (binary) | machine code (hex) |
| .text | li $r1, 6 | 1101 0000 0000 0110 | d006 |
|  | li $r2, 4 | 1101 0000 0100 0100 | d044 |
|  | addi $r3, $r1, 5 | 0001 0000 1000 0101 | 1085 |
|  | andi $r4, $r2, 3 | 0010 0010 1100 0011 | 22c3 |
|  | ori $r5, $r3, 6 | 0011 0101 0000 0110 | 3506 |
|  | li $r1, 2 | 1101 0000 0000 0010 | d002 |
|  | lsl $r5, $r5, $r1 | 0000 1000 0010 0100 | 0824 |
|  | li $r1, 3 | 1101 0000 0000 0011 | d003 |
|  | lsr $r6, $r5, $r1 | 0000 1000 0010 1101 | 082d |
|  | neg $r7, $r6 | 0000 1010 0011 0111 | 0a37 |
|  | li $r1, 1 | 1101 0000 0000 0001 | d001 |
|  | asr $r8, $r7, $r1 | 0000 1100 0011 1110 | 0c3e |
|  | max $r5, $r3, $r4 | 0101 0100 1110 0000 | 54e0 |
|  | min $r6, $r7, $r8 | 0110 1101 1110 1000 | 6de8 |
|  | halt | 1110 0000 0000 0000 | e000 |

1. Please answer what is the value stored in each register after executing the program

|  |  |
| --- | --- |
| register | value |
| $r1 | 1 |
| $r2 | 4 |
| $r3 | 11 |
| $r4 | 0 |
| $r5 | 11 |
| $r6 | -7 |
| $r7 | -7 |
| $r8 | -4 |

**Q4:**

1. Please explain the format of each of the following instructions (including the format of this instruction and its operation & function code).

* jump

jump x

function: Jump to address (PC + x)

OP code: 1010

Format of instruction: 1010 x

* call

cal x

function: Jump to address (PC + x), where x is an immediate number, and at the same time save (PC+1) to a special register $ra ($ra is not one of the 8 general registers $r1 -- r8)

OP code: 1011

Format of instruction: 1011 x

* rtn

rtn

function: Jump to address stored in $ra.

OP code: 0111

Format of instruction: 0111 0000 0000 0000

1. Please fill the following table with the binary machine code of each instruction of the testing program:

|  |  |  |  |
| --- | --- | --- | --- |
| label | instruction | machine code (binary) | machine code (hex) |
| .text | li $r1, 6 | 1101 0000 0000 0110 | d006 |
|  | li $r2, 4 | 1101 0000 0100 0100 | d044 |
|  | call 7 | 1011 0000 0000 0111 | b007 |
|  | move $r5, $r4 | 1111 0111 0000 0000 | f700 |
|  | li $r1, 7 | 1101 0000 0000 0111 | d007 |
|  | li $r2, 8 | 1101 0000 0100 1000 | d048 |
|  | call 3 | 1011 0000 0000 0011 | b003 |
|  | move $r6, $r4 | 1111 0111 0100 0000 | f740 |
|  | jump 4 | 1010 0000 0000 0100 | a004 |
|  | add $r3, $r1, $r2 | 0000 0000 0101 0000 | 0050 |
|  | mul $r4, $r1, $r3 | 0000 0000 1001 1011 | 009b |
|  | rtn | 0111 0000 0000 0000 | 7000 |
|  | halt | 1110 0000 0000 0000 | e000 |

1. Please answer what is the value stored in each register after executing the program

|  |  |
| --- | --- |
| register | value |
| $r1 | 7 |
| $r2 | 8 |
| $r3 | 15 |
| $r4 | 105 |
| $r5 | 60 |
| $r6 | 105 |
| $r7 | 0 |
| $r8 | 0 |