ELEC3441 Lab2

Jiang Feiyu 3035770800

Checkoff 1

Trace through the execution of the code and compare the machine instructions and their corresponding source assembly instructions. Then answer the following:

(i) How many machine instructions have the li instruction been assembled into?

Accoding to the `rars` program, the text segment is shown as following:

```
addi x11, x0,23
lui x12,0x0001234b
addi x12, x12,0xfffffbcd
add x10, x11, X12
```

There are 4 machine instructions in total, and the `li` instruction is assambled into **two machine instructions**.

(ii) Experiment with loading different constants. How are the li instructions being assembled differently with different constant values?

I tried to load with different constants, like `2047` (11 bits of binary), `2048` (12 bits of binary), `65535` (16 bits of binary)and here is the findings:

1. `li   a1, 2047`

   ```
   addi x11, x0,0x000007ff
   lui x12,0x0001234b
   addi x12,x12,0xfffffbcd
   add X10, x11, X12
   ```

2. `li   a1, 2048`

```
lui x11,1
addi x11,X11,0xfffff800
lui x12,0x0001234b
addi x12, x12,0xfffffbcd
add x10, x11, X12
```

3. `li   a1, 65535`

```
lui x11,16
addi x11, x11, 0
lui x12,0x0001234b
addi x12, x12,0xfffffbcd
add x10, x11, x12
```

- If the immediate value is more or equal to 12 bits, it will be split into two machine instructions: `lui` and `addi`, otherwise there will be one `addi` instruction only.

Checkoff 2

Trace through the execution of the file `loadstore.s` and answer the following questions:

1.What are the values in the array arrayPrime after the code has completed?

{11,13,24,19}

- Initial array values: 11,13,17,19
- First, the code loads the first two words (each word being 4 bytes) from the base address 0x10010000. The `lw` instruction loads `11` into register `t0` and `13` into register `t1`.
- Then, it executes the addition instruction `add`, adding the values in `t0` and `t1`, resulting in 11+13=24, which is stored in register `t2`.
- Finally, it stores the value `24` from register `t2` back into the location of the third element of the array, replacing the original `17`.
- Therefore, the array becomes: 11,13,24,1911,13,24,19.

2.What is the base address of arrayPrime?

0x10010000

3.If you insert another .word before the label arrayPrime and assemble the file again, where would the values of arrayPrime be moved to in memory?

Before insert another `.word`, we may get the value of location from RARS. At the address of `0x10010000`, the value is `b` for `+0`, `d` for `+4`, `11` for `+8` and `13` for `+c`. Thus, the value of the arrayPrime which is `{11, 13, 17 and 19}` has been saved in the the location start from `0x10010000` in the memory.

Bkpt	Address	Code	Basic	Source
	0x00400000	0x100105b7	lui x11,0x00010010	7: li a1, 0x10010000 # Hack alert. ...
	0x00400004	0x00058593	addi x11,x11,0	
	0x00400008	0x0005a283	lw x5,0(x11)	8: lw t0, 0(a1)
	0x0040000c	0x0045a303	lw x6,4(x11)	9: lw t1, 4(a1)
	0x00400010	0x006283b3	add x7,x5,x6	10: add t2, t0, t1
	0x00400014	0x0075a423	sw x7,8(x11)	11: sw t2, 8(a1)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x0000000b	0x0000000d	0x00000011	0x00000013	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

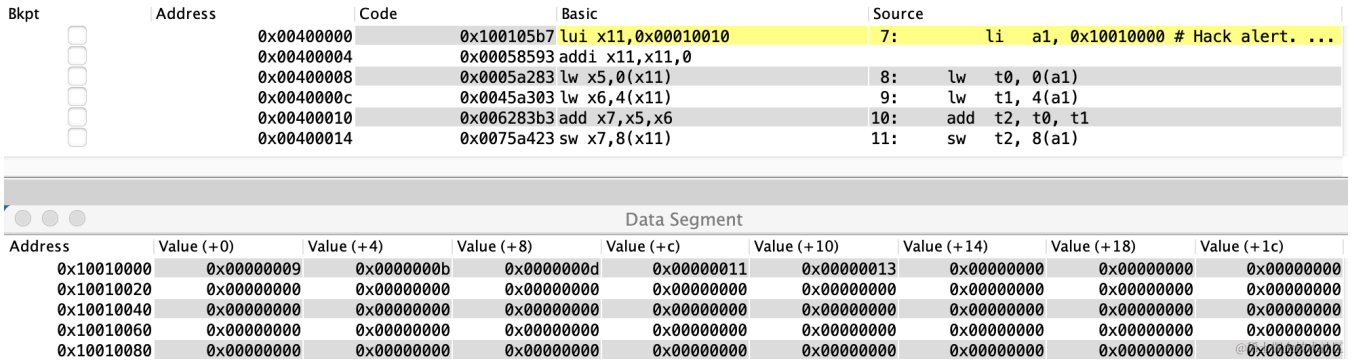
If we insert another .word before the label, the arrayPrime would be like:

```
.globl _start
.text
_start:
        li   a1, 0x10010000 # Hack alert.  Only works with rars
    lw   t0, 0(a1)
    lw   t1, 4(a1)
    add  t2, t0, t1
    sw   t2, 8(a1)

.data
.word 9
arrayPrime:
.word 11
.word 13
.word 17
.word 19
```

For the address of `0x00400000`, the value starts from `9` and same as the `.word` insert. And all the elements in the array are **shifted four bits** backward in the memory.

E.g.: `11` for `0x10010000` (value +4) and `13` for `0x10010000` (value +8)



Bkpt	Address	Code	Basic	Source
	0x00400000	0x100105b7	lui x11,0x00010010	7: li a1, 0x10010000 # Hack alert. ...
	0x00400004	0x00058593	addi x11,x11,0	
	0x00400008	0x0005a283	lw x5,0(x11)	8: lw t0, 0(a1)
	0x0040000c	0x0045a303	lw x6,4(x11)	9: lw t1, 4(a1)
	0x00400010	0x006283b3	add x7,x5,x6	10: add t2, t0, t1
	0x00400014	0x0075a423	sw x7,8(x11)	11: sw t2, 8(a1)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000009	0x0000000b	0x0000000d	0x00000011	0x00000013	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Checkoff 3

(i) What is the value printed at the Run I/O pane after running the program?

```
The count is: 3
```

(ii) In word, describe what is the function of the program mystery.s?

This code is an assembler that counts the number of odd numbers in an array and prints the result, which performs a bitwise AND operation on the input number with the binary value "0001", which checks if the least significant bit is set to 1. If it is, then the number is odd and the function returns a value of 1.