Homework 2

A.7

a. the code for MIPS:

DADD R1, R0, R0 ; i=0

SD 7000(R0), R1 ; store i

loop: LD R1, 7000(R0) ; load i

DSLL R2, R1, #3

LD R3, 3000(R2) ; load B[i]

LD R4, 5000(R0) ; load C

DADD R5, R3, R4 ; B[i] + C

SD 1000(R2), R5 ; A = B[i] + C

DADDI R1, R1 ,#1 ; i +=1

SD 7000(R0), R1 ; store i

DADDI R6,R1,#-101 ; compare

BNEZ R6,loop ; continue loop

The number of instructions executed dynamically: 2 + ( 10 \*101 ) = 1012

Memory-data references executed: 1 + 5 \* 101 = 506

Instruction bytes: 4 \* 12 = 48

b. the code for x86:

instruction length：

movq $0x0, %rax ; I = 0 3

movq %rax, 0x1b58(%rax) ; store I 4

movq $0x0, %rbp ; rbp = 0 3

loop: movq 0x1b58(%rbp), %rax 4

shl $0x3, %rdx ; load I 3

movq 0x0bb8(%rdx), %rbx ; load B[i] 4

movq 0x1388(%rbp), %rcx ; load C 4

add %rbx, %rcx ; B[i] + C 2

movq %rcx, 0x03e8(%rdx) ; A = B[i] + C 4

add $0x1, %rax ; i +=1 3

movq %rax, 0x1b58(%rbp) ; store I 4

cmpq $0x0065, %rax ; compare 4

jae loop ; continue loop 2

The number of instructions executed dynamically: 3 + ( 10 \*101 ) = 1013

Memory-data references executed: 1 + 5 \* 101 = 506

Instruction bytes: 44

A 18:

a.

1、Accumulator:

Load B p, m

Add C p, m

Store A p, m

Add C p, m // the result is passed to another instruction as an operand,

// loaded from memory after having been loaded once

Store B p, m

Minus p // the result is passed to another instruction as an operand,

Add A p, m // loaded from memory after having been loaded once

Store D p, m

2、Memory-memory:

Add A, B, C m

Add B, A, C m // loaded from memory after having been loaded once

Sub D, A, B m // loaded from memory after having been loaded once

3、Stack:

Push B p, m

Push C p, m

Add p

Pop A p, m

Push A p, m // loaded from memory after having been loaded once

// the result is passed to another instruction as an operand,

Push C p, m // loaded from memory after having been loaded once

Add p

Pop B p, m

Push B p, m // loaded from memory after having been loaded once

// the result is passed to another instruction as an operand,

Push A p, m // loaded from memory after having been loaded once

Sub p

Pop D p, m

4、Load-store:

Load R1, B p, m

Load R2, C p, m

Add R3, R1, R2 p

Store A, R3 p, m

Add R1, R3, R2 p // the result is passed to another instruction as an operand,

Store B, R1 p, m

Sub R4, R3, R1 p // the result is passed to another instruction as an operand,

Store D, R4 p, m

b. ( mark in the code above )

storage within the processor: marked as p;

storage in memory: marked as m

c.

1、Accumulator:

Add/load/store: 3bytes \*7

Minus: 1bytes \*1

instruction bytes are fetched: 22

bytes of data are transferred from/to memory: 7\*2 = 14bytes

code + data = 36bytes

2、Memory-memory::

Add: 7bytes \*3

instruction bytes are fetched: 21

bytes of data are transferred from/to memory: 9\*2 = 18bytes

code + data = 39bytes

3、Stack:

Push/pop: 3bytes \*9

Add/sub: 1bytes \*3

instruction bytes are fetched: 30

bytes of data are transferred from/to memory: 9\*2 = 18bytes

code + data = 48bytes

4、Load-store:

16 general-purpose registers => each 4 bits

Load/store; 4bytes \* 5

Add/sub: 3bytes \*3

instruction bytes are fetched: 29

bytes of data are transferred from/to memory: 5\*2 = 10bytes

code + data = 39bytes

* **Accumulator architecture is most efficient as measured by total memory traffic**

d.

1、Accumulator:

Add/load/store: 9bytes \*7

Minus: 1bytes \*1

instruction bytes are fetched: 64

bytes of data are transferred from/to memory: 7\*8 = 56bytes

code + data = 120bytes

2、Memory-memory:

Add: 25bytes \*3

instruction bytes are fetched: 75

bytes of data are transferred from/to memory: 9\*8 = 72bytes

code + data = 147bytes

3、Stack:

Push/pop: 9bytes \*9

Add/sub: 1bytes \*3

instruction bytes are fetched: 82

bytes of data are transferred from/to memory: 9\*8 = 72bytes

code + data = 154bytes

4、Load-store:

16 general-purpose registers => each 4 bits

Load/store; 10bytes \* 5

Add/sub: 3bytes \*3

instruction bytes are fetched: 59

bytes of data are transferred from/to memory: 5\*8 = 40bytes

code + data = 99bytes

* Load-store architecture becomes the most efficient as measured by total memory traffic;
* Memory-memory performs worse than before;
* Accumulator architecture is not as efficient as before due to the loading from memory after having been loaded once
* Stack is the worst as before.