Sourcecode - **AccumSim**

.data

0x00300000:3

0x00300001:7

0x00300002:5

0x00300003:4

0x00300004:0

0x00300005:0

.text

LOAD 0x00300000

MULT 0x00300000

MULT 0x00300001

STORE 0x00300004

LOAD 0x00300002

MULT 0x00300000

ADD 0x00300004

ADD 0x00300003

END

Sourcecode -**Stacksim**

.data

0x00300000:3

0x00300001:7

0x00300002:5

0x00300003:4

.text

PUSH 0x00300001

PUSH 0x00300000

PUSH 0x00300000

MULT

MULT

PUSH 0x00300002

PUSH 0x00300000

MULT

ADD

PUSH 0x00300003

ADD

END

Binary translation

.data

0x0030000000000003

0x0030000100000007

0x0030000200000005

0x0030000300000004

0x0030000400000000

0x0030000500000000

.text

0xE100300000

0x0F00300000

0x0F00300001

0x7700300004

0xE100300002

0x0F00300000

0xAA00300004

0xAA00300003

A5

.data

0x0030000000000003

0x0030000100000007

0x0030000200000005

0x0030000300000004

text

0xFF00300001

0xFF00300000

0xFF00300000

0F

0F

0xFF00300002

0xFF00300000

0F

AA

0xFF00300003

AA

A5

ADD 0b10101010 -> HEX -> AA

SUB 0b01010101 -> HEX -> 55

MULT 0b00001111 -> HEX -> F

DIV 0b11110000 -> HEX -> F0

PUSH 0b11111111 -> HEX -> FF

POP 0b00000000 -> HEX -> 0

LOAD 0b11100001 -> HEX -> E1

STORE 0b01110111 -> HEX -> 77

END 0b10100101 -> HEX -> A5

(Data seg number \* bits/mem\_add + text\_seg \* bits/instr) / 8

(6 \* 32 + 9 \* 32 )/8 = **60 bytes**

MIPS:

(4 \* 32 + 9 \* 32) / 8 = **52 bytes**

(Data seg number \* bits/mem\_add + text\_seg \* bits/instr) / 8

(6 \* 32 + 9 \* 32) / 8 = **60 bytes**

|  |  |  |
| --- | --- | --- |
| Instructions | Opcode (8 bits) | Address (24-bits) |
| PUSH | 0xFF | Source |
| POP | 0x00 | Destination |
| MULT | 0x0F | N/A |
| ADD | 0xAA | N/A |
| SUB | 0x55 | N/A |
| DIV | 0xF0 | N/A |
| END | 0xA5 | N/A |

|  |  |  |
| --- | --- | --- |
| Instructions | Opcode (8 bits) | Address (24-bits) |
| LOAD | 0xE1 | Source |
| STORE | 0x77 | Destination |
| MULT | 0x0F | Source |
| ADD | 0xAA | Source |
| END | 0xAF | N/A |

Hypothetic encoding from part #4

Differences between types of instructions: Opcodes with a 1 as the most significant bit is a immediate operand type, and with a 0 as the most significant bit will be of the memory address operand type.

Types:

|  |  |
| --- | --- |
| 8 bit op code | 32 bit mem\_addr |
| 8 bit op code | 32 bit signed |

Push: 0x006FFFFFFF

Pop: 0x016FFFFFFF

Add: 0x026FFFFFFF

Mult: 0x036FFFFFFF

PushS: 0x1000000001

Load: 0x006FFFFFFF

Store: 0x016FFFFFFF

Add: 0x026FFFFFFF

Mult: 0x036FFFFFFF

LoadS: 0x1000000001