***Opcode Tables***

|  |  |  |  |
| --- | --- | --- | --- |
| **Stack base machine** | |  |  |
| Mnemonic | Opcode | Effect | Comments |
| PUSH, m | 00000005 | Top of stack <== (m..m+7) | Place contents in memory to Top of Stack |
| POP, m | 00000006 | Top of stack ==> (m…m+7) | Place contents from Top of Stack to memory |
| ADD | 00000007 | Top of stack + next on stack ==> Top of stack | Add Top of Stack and Next on Stack and place result on Top of Stack |
| MULT | 00000008 | Top of stack \* next on stack ==> Top of stack | Multiply Top of Stack and Next on Stack and place result on Top of Stack |
| END | 0000FFFF | Terminate Program |  |
| ADDI, n | 00000010 | Top of stack + (number) ==> Top of stack | Add an immediate value and the Top of Stack and place result on Top of Stack |
| MULI ,n | 00000011 | Top of stack \* (number) ==> Top of stack | Multiply an immediate value and the Top of Stack and place result on Top of Stack |
|  |  |  |  |
| **Accumulator base machine** | | |  |
| Mnemonic | Opcode | Effect | Comments |
| LOAD, m | 00000009 | [A] <== (m…m+7) | Place contents in memory into Accumulator Register |
| STO, m | 00000010 | [A] ==> (m…m+7) | Place contents in Accumulator Register into memory |
| ADD, m | 00000007 | [A] <== [A] + (m…m+7) | Add Accumulator Register and contents in memory and place result in Accumulator Register |
| MULT, m | 00000008 | [A] <== [A] \* (m…m+7) | Multiply Accumulator Register and contents in memory and place result in Accumulator Register |
| END | 00000020 | Terminate Program |  |
| LODI,n | 00000021 | [A] <== (number) | Place an immediate value into the Accumulator Register |
| ADDI,n | 00000022 | [A] <== [A] + (number) | Add Accumulator Register and an immediate number and place result in Accumulator Register |

**Accumulator Base Machine Code**

.text Instruction Code

LOAD, A 0x0000 0009 1000 0000

MULT, X 0x0000 0008 1000 00C0

MULT, X 0x0000 0008 1000 00C0

STO, A 0x0000 0010 1000 0000

LOAD, B 0x0000 0009 1000 0040

MULT, X 0x0000 0008 1000 00C0

STO, B 0x0000 0010 1000 0040

LOAD, C 0x0000 0009 1000 0080

ADD, B 0x0000 0007 1000 0040

ADD, A 0x0000 0007 1000 0000

STO, C 0x0000 0010 1000 0080

END 0x0000 0020 0000 0000

.data Value Memory Addresses:

A: 3 0x0000 0000 0000 0003 0x1000 0000

B: 4 0x0000 0000 0000 0004 0x1000 0040

C: 5 0x0000 0000 0000 0005 0x1000 0080

X: 5 0x0000 0000 0000 0005 0x1000 00C0

***96 bytes instructions***

***32 bytes data***

***96 bytes + 32 bytes = 136 bytes total***

**Stack Base Machine Code**

.text Instruction Code

PUSH, A 0x0000 0005 1000 0000

PUSH, X 0x0000 0005 1000 00C0

PUSH, X 0x0000 0005 1000 00C0

MULT 0x0000 0008 0000 0000

MULT 0x0000 0008 0000 0000

PUSH, B 0x0000 0005 1000 0040

PUSH, X 0x0000 0005 1000 00C0

MULT 0x0000 0008 0000 0000

PUSH, C 0x0000 0005 1000 0080

ADD 0x0000 0007 0000 0000

ADD 0x0000 0007 0000 0000

POP, X 0x0000 0006 1000 00C0

END 0x0000 FFFF 0000 0000

.data Value Memory Location

A: 3 0x0000 0000 0000 0003 0x1000 0000

B: 4 0x0000 0000 0000 0004 0x1000 0040

C: 5 0x0000 0000 0000 0005 0x1000 0080

X: 5 0x0000 0000 0000 0005 0x1000 00C0

***104 bytes instructions***

***32 bytes data***

***104 bytes + 32 bytes = 136 bytes***

**MIPS Machine**

Text Segment

=========================

[0x00400000] 0x8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp)

[0x00400004] 0x27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 .

.

.

.

0x00400084] 0x3402000a ori $2, $0, 10 ; 36: li $v0, 10 # g'bye

[0x00400088] 0x0000000c syscall ; 37: syscall

***0x0040008C - 0x00400000 = 8C = 140 Bytes***

Data Segment

=========================

DATA

[0x10000000]...[0x10010000] 0x00000000

[0x10010000] 0x00000003 0x00000007 0x00000005 0x00000004

[0x10010010] 0x77736e41 0x3d207265 0x000a0020 0x00000000

[0x10010020]...[0x10040000] 0x00000000

***0x10010020 - 0x10010000 = 20 = 32 bytes***

***140 bytes for Instructions***

***32 bytes for Data***

***140 bytes + 32bytes = 172 bytes***