TABLES

|  |  |
| --- | --- |
| **Opcode Name** | **Binary Value** |
| ADD | 000000 |
| ADD2 | 000001 |
| SUBTRACT | 000010 |
| SUBTRACT2 | 000011 |
| MULTIPLY | 000100 |
| MULTIPLY2 | 000101 |
| DIVIDE | 000110 |
| DIVIDE2 | 000111 |
| MOVE | 001000 |
| JUMPTO | 001001 |
| CALL | 001010 |
| STACK | 001011 |
| RETURN | 010110 |
| POP | 001100 |
| POPVAL | 001101 |
| COMPARE | 001110 |
| COMPARE\_GT | 001111 |
| COMPARE\_LT | 010000 |
| COMPARE\_GE | 010001 |
| COMPARE\_LE | 010010 |
| COMPARE\_EQ | 010011 |
| BRANCH\_AND | 010100 |
| BRANCH | 010101 |

REGISTERS

|  |  |  |
| --- | --- | --- |
| **Name** | **Binary Value** | **Explanation** |
| GP1 | 0000 | One of four general purpose registers |
| GP2 | 0001 | One of four general purpose registers |
| GP3 | 0010 | One of four general purpose registers |
| GP4 | 0011 | One of four general purpose registers |
| AR1 | 0100 | One of four general purpose registers specially named for procedure arguments |
| AR2 | 0101 | One of four general purpose registers specially named for procedure arguments |
| AR3 | 0110 | One of four general purpose registers specially named for procedure arguments |
| AR4 | 0111 | One of four general purpose registers specially named for procedure arguments |
| RV1 | 1000 | One of four general purpose registers specially named for values being returned from the a procedure |
| RV2 | 1001 | One of four general purpose registers specially named for values being returned from the a procedure |
| RV3 | 1010 | One of four general purpose registers specially named for values being returned from the a procedure |
| RV4 | 1011 | One of four general purpose registers specially named for values being returned from the a procedure |
| SP | 1100 | The Stack Pointer. Holds the address of the top of the stack |
| TF | 1101 | The comparison register (we call it the True/False register). Holds the results of various comparison operations |
| PC | 1110 | The Program Counter |

ADDRESS MODES

|  |  |  |
| --- | --- | --- |
| **Name** | **Binary Value** | **Description** |
| REGISTER | 000 | Accessing a register |
| DIRECT | 001 | Accessing a memory value directly |
| INDEX | 010 | Accessing a memory value with an index offset |
| INDIRECT | 011 | Accessing something like a memory's offset or a pointer |
| IMMEDIATE | 100 | Accessing a direct value that is hard-coded into the instruction; a literal like 5 or 'a' |