MIPS Functional Verification

**Perform Multiplication of 6 and 5 without MUL instruction**

|  |  |  |  |
| --- | --- | --- | --- |
| Inst\_mem\_address | Assembly Instruction | Binary inst code | Comments |
| 0 | Mov R0, #5; | 00001\_00000\_00000\_1\_0000\_0000\_0000\_0101 | Set Register, R0=5 |
| 1 | Mov R1, #6; | 00001\_00001\_00000\_1\_0000\_0000\_0000\_0110 | R1=6 |
| 2 | Mov R2, #0; | 00001\_00010\_00000\_1\_0000\_0000\_0000\_0000 | R2=0 |
| 3 | Mov R3, #6; | 00001\_00011\_00000\_1\_0000\_0000\_0000\_0110 | R3=6 |
| 4 | ADD R2,R2,R0 | 00010\_00010\_00010\_0\_0000\_0000\_0000\_0000 | Final value of R2=30 |
| 5 | SUB R3,R3, #1; | 00011\_00011\_00011\_1\_0000\_0000\_0000\_0001 | R3=5,4,3,…,0 |
| 6 | JNZ @4 | 11000\_00000\_00000\_0\_0000\_0000\_0000\_0100 | Jump to inst\_mem\_addr 4 when ‘jumpnozero is high |
| 7 | Mov R4,R2 | 00001\_00100\_00010\_0\_0000\_0000\_0000\_0000 | Final value of R4=30 |
| 8 | HALT | 11011\_00000\_00000\_0\_0000\_0000\_0000\_0000 |  |
|  |  |  |  |

R3=0

**Simulated Wave Form**

R4=30

R2=30

R1=6

R0=5

A screenshot of a computer

Description automatically generated

**Appendix**

-----------------------------------------------------------------------------------------------------------------------------------------

**Instruction Format for the Instruction Register**

IR <--ir[31:27]--><--ir[26:22]--><--ir[21:17]--><--ir[16]--><--ir[15:11]--><--ir[10:0]-->

Fields <--- oper --><-- rdest --><-- rsrc1 --><--modesel--><-- rsrc2 --><--unused -->

Fields <--- oper --><-- rdest --><-- rsrc1 --><--modesel--><-- immediate\_date -->

-----------------------------------------------------------------------------------------------------------------------------------------

**OP Codes for different operation types**

+----------------+---------------------+

| Arithmetic | Op code |

+----------------+---------------------+

| movsgpr | 5'b00000 |

| mov | 5'b00001 |

| add | 5'b00010 |

| sub | 5'b00011 |

| mul | 5'b00100 |

+----------------+---------------------+

+----------------+---------------------+

| Logical | Op code |

+----------------+---------------------+

| or | 5'b00101 |

| and | 5'b00110 |

| xor | 5'b00111 |

| xnor | 5'b01000 |

| nand | 5'b01001 |

| nor | 5'b01010 |

| not | 5'b01011 |

+----------------+---------------------+

+----------------+---------------------+

| Load & Store | Op code |

+----------------+---------------------+

| storereg| 5'b01101 |

| storedin| 5'b01110 |

| senddout| 5'b01111 |

| sendreg | 5'b10001 |

+----------------+---------------------+

+----------------+---------------------+

| Jump and Branch | Op code |

+----------------+---------------------+

| jump | 5'b10010 |

| jcarry | 5'b10011 |

| jnocarry| 5'b10100 |

| jsign | 5'b10101 |

| jnosign | 5'b10110 |

| jzero | 5'b10111 |

| jnozero | 5'b11000 |

| joverflow| 5'b11001 |

| jnooverflow| 5'b11010 |

+----------------+---------------------+

+----------------+---------------------+

| Halt | Op code |

+----------------+---------------------+

| halt | 5'b11011 |

+----------------+---------------------+