branchingCPU

# Kian Holden

This machine has **32 general purpose registers**. **Each register can store 8 bytes**, and data memory is byte addressed with 2^16 total bytes. The machine is little endian. Register 31 is the zero register and register 30 is the link register. X0 is the LEDoutput register (this means that when the program halts, X0 is displayed on the LED). All instructions are 32 bits.

To use the machine:

Write your code as a series of the above instructions in the .txt file named ‘project2code.txt’ (see the instruction set architechture at the bottom of this document). Note that commas and square brackets are all optional, but **spaces between the mnemonics and operands are required, and each instruction must be on its own line. Labels must be in the form ‘labelName:’. There must not be whitespace before the colon, and the label must only include alphanumerics (no spaces).** You can refer to registers as ‘Xn’ or ‘xn’, where n is a number between 0 and 31. Make sure to put the output in register 0 (for the LEDoutput) and to use the “STOP” command at the end of the program.

To run the assembler, go to the terminal, navigate to the directory with all the files, and run the following command

$ python .\project2Assemblerv2.py

This worked on my machine with python 3.1.1 installed

After that,

1. open the circ file
2. navigate to the subcircuit called ‘InstructionMemory’
3. reset the simulation (ctrl + r)
4. right click on the RAM in ‘InstructionMemory’
5. click ‘load image’ and select ‘v2InstImage.txt’
6. navigate to the subcircuit called ‘DataMemory’
7. clock ‘load image’ and select ‘v2DataImage.txt’
8. Then you can simulate the CPU by moving forward through clock cycles.

|  |  |
| --- | --- |
| Instruction | description |
| MOV Rd, Rn, Rm | Rm into Rd |
| ORR Rd, Rn, Rm | bitwise or with Rn and Rm into Rd |
| EOR Rd, Rn, Rm | bitwise exclusive or with Rn and Rm into Rd |
| AND Rd, Rn, Rm | bitwise and with Rn and Rm into Rd |
| ADD Rd, Rn, Rm | sum of Rn and Rm into Rd |
| SUB Rd, Rn, Rm | difference of Rn and Rm into Rd |
| MUL Rd, Rn, Rm | product of Rn and Rm into Rd |
| SDIV Rd, Rn, Rm | Signed integer division of Rn and Rm into Rd |
| UDIV Rd, Rn, Rm | Unsigned integer division of Rn and Rm into Rd |
| LSL Rd, Rn, Rm | Logical left shift of Rn by Rm into Rd |
| LSR Rd, Rn, Rm | Logical right shift of Rn by Rm into Rd |
| ASR Rd, Rn, Rm | Arithmetic right shift of Rn by Rm into Rd |
| ANDS Rd, Rn, Rm | Bitwise and of Rn and Rm into Rd. Sets condition codes. |
| ADDS Rd, Rn, Rm | Sum of Rn and Rm into Rd. Sets condition codes. |
| SUBS Rd, Rn, Rm | Difference of Rn and Rm into Rd. Sets condition codes. |
| CMP Rn, Rm | Sets condition codes based on difference of Rn and Rm. |
| LDRB Rt, [Rn, Rm] | Loads 1 byte of memory from address (Rn+Rm) into Rt |
| LDRH Rt, [Rn, Rm] | Loads 2 bytes of memory from address (Rn+Rm) into Rt |
| LDRW Rt, [Rn, Rm] | Loads 4 bytes of memory from address (Rn+Rm) into Rt |
| LDR Rt, [Rn, Rm] | Loads 8 bytes of memory from address (Rn+Rm) into Rt |
| STRB Rt, [Rn, Rm] | Stores least significant byte of Rt into memory at address (Rn+Rm) |
| STRH Rt, [Rn, Rm] | Stores least significant 2 bytes of Rt into memory at address (Rn+Rm) |
| STRW Rt, [Rn, Rm] | Stores least significant 4 bytes of Rt into memory at address (Rn+Rm) |
| STR Rt, [Rn, Rm] | Stores all 8 bytes of Rt into memory at address (Rn+Rm) |
|  | |
| MOV Rd, 11simm | Puts the 11 bit signed immediate into Rd |
| ORR Rd, Rn, 11simm | Puts bitwise or with Rn and 11bit signed immediate into Rd (sign extends immediate) |
| EOR Rd, Rn, 11simm | Puts bitwise exclusive or with Rn and 11bit signed immediate into Rd (sign extends immediate) |
| AND Rd, Rn, 11simm | Puts bitwise and with Rn and 11bit signed immediate into Rd (sign extends immediate) |
| ADD Rd, Rn, 11simm | sum of Rn and the 11 bit signed immediate into Rd |
| SUB Rd, Rn, 11simm | difference of Rn and the 11 bit signed immediate into Rd |
| MUL Rd, Rn, 11simm | product of Rn and the 11 bit signed immediate into Rd |
| SDIV Rd, Rn, 11simm | Signed division of Rn and 11 bit signed immediate into Rd |
| UDIV Rd, Rn, 11simm | Unsigned division of Rn and 11 bit signed immediate into Rd |
| LSL Rd, Rn, 6imm | Logical left shift of Rn by 6 bit immediate into Rd |
| LSR Rd, Rn, 6imm | Logical right shift of Rn by 6 bit immediate into Rd |
| ASR Rd, Rn, 6imm | Arithmetic right shift of Rn by 6 bit immediate into Rd |
| LDRB Rt, [Rn, 11simm] | Loads 1 byte of memory from (Rn+11simm) into Rt. Sign extends the byte to 8 bytes |
| LDRH Rt, [Rn, 11simm] | Loads 2 bytes of memory from (Rn+11simm) into Rt. Sign extends the bytes to 8 bytes. |
| LDRW Rt, [Rn, 11simm] | Loads 4 bytes of memory from (Rn+11simm) into Rt. Sign extends the bytes to 8 bytes. |
| LDR Rt, [Rn, 11simm] | Loads 8 bytes of memory from (Rn+11simm) into Rt |
| STRB Rt, [Rn, 11simm] | Stores least significant byte of Rt at address (Rn+11simm) |
| STRH Rt, [Rn, 11simm] | Stores least significant 2 bytes of Rt at address (Rn+11simm) |
| STRW Rt, [Rn, 11simm] | Stores least significant 4 bytes of Rt at address (Rn+11simm) |
| STR Rt, [Rn, 11simm] | Stores all 8 bytes of Rt at address (Rn+11simm) |
|  | |
| B label | Branches to label |
| BL label | Sets link register to PC+1 and branches to label |
| B.EQ label | Branches to label if Z flag is 1 |
| B.NE label | Branches to label if Z flag is 0 |
| B.LT label | Branches to label if N != V |
| B.LO label | Branches to label if C flag is 0 |
| B.LE label | Branches to label if ~(Z == 0 && N==V) |
| B.LS label | Branches to label if ~(Z==0 && C == 1) |
| B.GT label | Branches to label if (Z == 0 && N==V) |
| B.HI label | Branches to label if ~(Z==0 && C == 1) |
| B.GE label | Branches to label if N==V |
| B.HS label | Branches to label if C is 1 |
| CBZ Rt, label | Branches to label if Rt is zero |
| CBNZ Rt, label | Branches to label if Rt is not zero |
| BR Rt | Branches to address of instruction given from register |
| RET | Sets PC to link register |
|  | |
| NOP | Does nothing (in reality this branches to the next instruction) |
| STOP | Stops the PC from incrementing (in reality this instruction branches to itself as a short infinite loop that does nothing). Also displays X0 in the LED component |

\*Note: all branching labels are pc-relative offsets calculated by the assembler

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instruction | Opcode | Operands | | | | |
| MOV Rd, Rm | Xx11xx00000 | Rd (5 bit) | 00000 | 111000 | | Rm (5 bit) |
| ORR Rd, Rn, Rm | Xx11xx00001 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| EOR Rd, Rn, Rm | Xx11xx00010 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| AND Rd, Rn, Rm | Xx11xx00011 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| ADD Rd, Rn, Rm | Xx11xx00100 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| SUB Rd, Rn, Rm | Xx11xx00101 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| MUL Rd, Rn, Rm | Xx11xx00110 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| SDIV Rd, Rn, Rm | Xx11xx00111 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| UDIV Rd, Rn, Rm | Xx11xx01000 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| LSL Rd, Rn, Rm | Xx11xx01001 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| LSR Rd, Rn, Rm | Xx11xx01010 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| ASR Rd, Rn, Rm | Xx11xx01011 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| ANDS Rd, Rn, Rm | Xx11xx01100 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| ADDS Rd, Rn, Rm | Xx11xx01101 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| SUBS Rd, Rn, Rm | Xx11xx01110 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| CMP Rn, Rm | Xx11xx01110 | 11111 | Rn (5 bit) | 111000 | | Rm (5 bit) |
| LDRB Rt, [Rn, Rm] | Xx01xxx0000 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| LDRH Rt, [Rn, Rm] | Xx01xxx0001 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| LDRW Rt, [Rn, Rm] | Xx01xxx0010 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| LDR Rt, [Rn, Rm] | Xx01xxx0011 | Rd (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| STRB Rt, [Rn, Rm] | Xx01xxx0100 | Rt (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| STRH Rt, [Rn, Rm] | Xx01xxx0101 | Rt (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| STRW Rt, [Rn, Rm] | Xx01xxx0110 | Rt (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
| STR Rt, [Rn, Rm] | Xx01xxx0111 | Rt (5 bit) | Rn (5 bit) | 111000 | | Rm (5 bit) |
|  | | | | | | |
| MOV Rd, 11simm | Xx11xx10000 | Rd (5 bit) | 00000 | 11 bit signed immediate | | |
| ORR Rd, Rn, 11simm | Xx11xx10001 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| EOR Rd, Rn, 11simm | Xx11xx10010 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| AND Rd, Rn, 11simm | Xx11xx10011 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| ADD Rd, Rn, 11simm | Xx11xx10100 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| SUB Rd, Rn, 11simm | Xx11xx10101 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| MUL Rd, Rn, 11simm | Xx11xx10110 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| SDIV Rd, Rn, 11simm | Xx11xx10111 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| UDIV Rd, Rn, 11simm | Xx11xx11000 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| LSL Rd, Rn, 6imm | Xx11xx11001 | Rd (5 bit) | Rn (5 bit) | 00000 | 6 bit unsigned imm | |
| LSR Rd, Rn, 6imm | Xx11xx11010 | Rd (5 bit) | Rn (5 bit) | 00000 | 6 bit unsigned imm | |
| ASR Rd, Rn, 6imm | Xx11xx11011 | Rt (5 bit) | Rn (5 bit) | 00000 | 6 bit unsigned imm | |
| ANDS Rd, Rn, 11simm | Xx11xx11100 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| ADDS Rd, Rn, 11simm | Xx11xx11101 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| SUBS Rd, Rn, 11simm | Xx11xx11110 | Rd (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| CMP Rn, 11simm | Xx11xx11110 | 11111 | Rn (5 bit) | 11 bit signed immediate | | |
| LDRB Rt, [Rn, 11simm] | Xx01xxx1000 | Rt (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| LDRH Rt, [Rn, 11simm] | Xx01xxx1001 | Rt (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| LDRW Rt, [Rn, 11simm] | Xx01xxx1010 | Rt (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| LDR Rt, [Rn, 11simm] | Xx01xxx1011 | Rt (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| STRB Rt, [Rn, 11simm] | Xx01xxx1100 | Rt (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| STRH Rt, [Rn, 11simm] | Xx01xxx1101 | Rt (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| STRW Rt, [Rn, 11simm] | Xx01xxx1110 | Rt (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
| STR Rt, [Rn, 11simm] | Xx01xxx1111 | Rt (5 bit) | Rn (5 bit) | 11 bit signed immediate | | |
|  | | | | | | |
| B label | Xx00xxx1100 | 00000 | 00000 | 11 bit signed immediate | | |
| BL label | Xx00xxx1101 | 11110  (Reg 30) | 00000 | 11 bit signed immediate | | |
| B.EQ label | Xx00xxx0000 | 00000 | 00000 | 11 bit signed immediate | | |
| B.NE label | Xx00xxx0001 | 00000 | 00000 | 11 bit signed immediate | | |
| B.LT label | Xx00xxx0010 | 00000 | 00000 | 11 bit signed immediate | | |
| B.LO label | Xx00xxx0011 | 00000 | 00000 | 11 bit signed immediate | | |
| B.LE label | Xx00xxx0100 | 00000 | 00000 | 11 bit signed immediate | | |
| B.LS label | Xx00xxx0101 | 00000 | 00000 | 11 bit signed immediate | | |
| B.GT label | Xx00xxx0110 | 00000 | 00000 | 11 bit signed immediate | | |
| B.HI label | Xx00xxx0111 | 00000 | 00000 | 11 bit signed immediate | | |
| B.GE label | Xx00xxx1000 | 00000 | 00000 | 11 bit signed immediate | | |
| B.HS label | Xx00xxx1001 | 00000 | 00000 | 11 bit signed immediate | | |
| CBZ Rt, label | Xx00xxx1010 | Rt (5 bit) | 00000 | 11 bit signed immediate | | |
| CBNZ Rt, label | Xx00xxx1011 | Rt (5 bit) | 00000 | 11 bit signed immediate | | |
| BR Rt | Xx00xxx1110 | Rt (5 bit) | 00000 | 00000000000 | | |
| RET | Xx00xxx1110 | 11110  (Reg 30) | 00000 | 00000000000 | | |
|  | | | | | | |
| NOP | Xx00xxx1100 | 000000000000000000001 | | | | |
| STOP | Xx00xxx1100 | 000000000000000000000 | | | | |