Shift Right Logical:

introduction:

MIPS also has a **shift right logical** instruction. It moves bits to the right by a number of positions less than 32. The high-order bit gets zeros and the low-order bits are discarded.

If the bit pattern is regarded as an unsigned integer, or a positive two's comp. integer, then a right shift of one bit position performs an integer divide by two. A right shift by N positions performs an integer divide by 2^N^.

the following is a machine code description for Srl:

```
srl \$rs \$rt shift
```

Recipe:

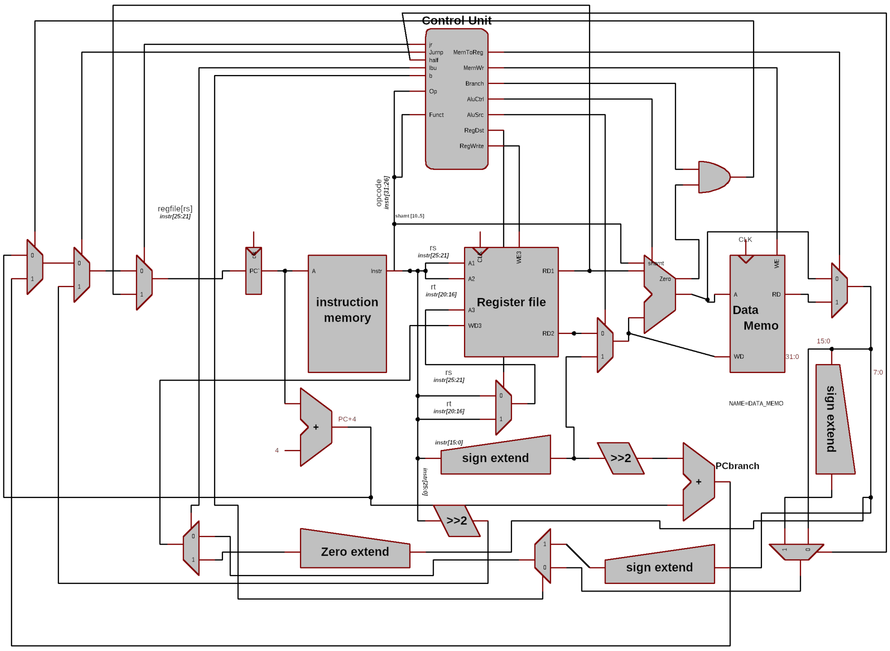
mux[] (multiplexer): It would select Read data 1(rs) if we're not doing a shift operation, and it would select(rt) if we are doing a shift operation.

branch Instruction: we would need to branch Instruction[10:6] (the shift amount) off of Instruction[15:0], and Instruction[10:6] would then be fed into the other port of the ALU

implementation:

option (shift)	operation
0	output of mux[1] equals not doing a shift
1	output of mux[1] equals doing a shift operation.

schmatic


Control Unit
jr
Jump
half
lbu
b
Op
Funct
MemToReg
MemWr
Branch
AluCtrl
AluSrc
RegDst
RegWrite
opcode
instr[31:26]
shamt [10..5]
regfile[rs]
instr[25:21]
PC'
CLK
A
Instr
instruction memory
rs
instr[25:21]
rt
instr[20:16]
A1
A2
A3
WD3
WE3
Register file
RD1
RD2
shamt
Zero
CLK
WE
A
RD
WD
Data Memo
31:0
15:0
7:0
NAME=DATA_MEMO
sign extend
rs
instr[25:21]
rt
instr[20:16]
PC+4
4
+
instr[15:0]
sign extend
>>2
+
PCbranch
instr[25:0]
>>2
Zero extend
sign extend
0
1