Load half and load byte

introduction:

a "Load half" and "Load byte" implementation using MIPS micro-architecture was built upon Harris design in their book (reference)

`lh $storeReg imm($regRefearingToMemAddress)`

`lb $storeReg imm($regRefearingToMemAddress)`

the following is a machine code description for lh and lb

```
lh: 100001 $regRefearingToMemAddress $storeReg iiiiiiii iiiiiiii
lb: 100000 $regRefearingToMemAddress $storeReg iiiiiiii iiiiiiii
```

Recipe:

Items/Pins:

1. pin_b (byte) : used as a selector for mux[2]
2. half (half-word) : used as a selector for mux[1]
3. mux[1] (multiplexer): a multiplexer provide an option to full word or half word
4. mux[2] (multiplexer): a multiplixer provide an option to mux[1] or one byte

implementation:

this design is based on the fact that `lw` was already implemented and working well so why not to reuse it? at the output of **MemToReg** multiplexer (`lw`'s output) i've used two multiplexers mux[1] and mux [2]

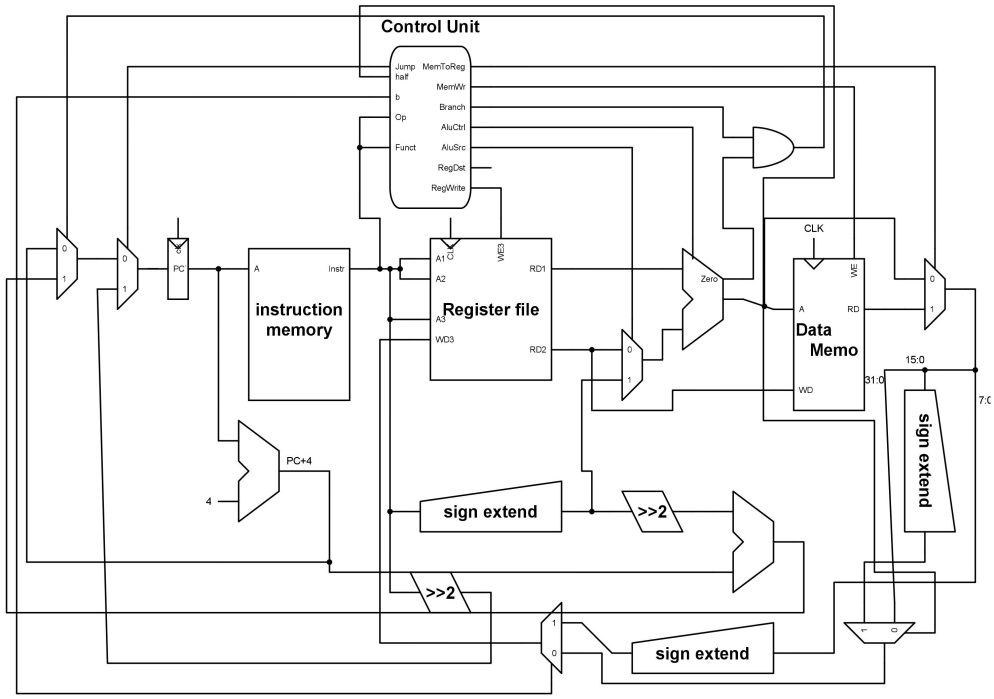
mux[1] will chose from the full word (32-bit) and a sign-extended half word `{16{halfword[15]},halfword[15:0]}` using **half pin** as a controller

option (half pin)	operation
0	output of mux[1] equals the full word
1	output of mux[1] equals half of the word

mux[2] will chose from mux[1] output and a sign-extended one byte `{24{8-bits[7]},8-bits[7:0]}` using **half pin** as a controller

option (b pin)	operation
0	output of mux[2] equals mux[1]
1	output of mux[2] equals sign extended one byte

schematic:



Code:

refearing to the diff file to make a quick review to what i've changed/added

Reference:

Digital design and computer architecture by David and Sarah Harris