Modules setup

1. PC: **input:** the computed newPC address, and its **output** is its current address.
2. Memory:
    a. Instruction memory: loads the binary machine code file (on initial only). **Input** received: the 8 bit binary address from PC. **Outputs:** input[4:0] goes to sign_extension(5to8), inst[7:6] goes to ALU data1 inst, inst[4] goes to a mux (input at 0) and ALUctrl, inst[7:5] goes to control unit directly, inst[1:0] goes to rt (read reg 1) of Register File; inst[3:2] goes to rs (read reg 2) and rs (write reg 2) of Register File and sign_extension(2to8).
    b. Data Memory: **input:** address computed from ALU, datawrite address from Register File, memWrite, memRead (from control unit). **Output:** register value as input to a mux
3. Control unit: **input:** inst[7:5] and inst[4]. **Output:** jctrl, jrctrl, memWrite, memRead, memToReg, ALUop, ALUsrc, regWrite, brctrl, ractrl.
4. Register File: **input:** regWrite, brctrl, ractrl (from control unit), rt (read reg 1) (inst[1:0]), rs (read reg 2) (inst[3:2]), rs (write reg 2) (inst[3:2]), datawrite. **Output:** rt (read data 1), rs (read data 2), raAddress
5. ALUctrl: **input:** ALUop (From control unit), inst[4] from instruction memory. **Output:** ALU(with ctrl) (1bit)
6. Mux: take in 8 bit inputs, 1 bit control, output 8 bit
7. ALU(with ctrl): 3-bit ALUop from control, take in two 8 bit inputs, 1-bit bit (equal) at 0 output, 8-bit at result output
8. ALU(regular): take in two 8 bitinputs, 1-bit (equal) at 0 output, 8-bit output at result output
9. Sign_extension: 1. 2to8 and 2. 5to8