Chapter 6 Handout Notes
10/13/08 Version 3.0

- Pipelining
  - Pipelining the control unit allows the execution of multiple instructions to be overlapped.
  - Instructions are overlapped to improve effective execution time.
  - Key factor in making today's computers faster.
  - See Figure 6.1 Laundry example.
  - A pipeline consists of several stages or segments.
    - Break instruction executions into steps (Book uses 5 - 4 or 5 typical some computers have more or less)
      - IF - Instruction fetch
      - ID - Instruction decode
      - EX - Execute (perform required ALU operation)
      - DM - Memory (access data memory during load or store)
      - WB - Write result into GPR
    - Ideally the throughput (how often an instruction exits the pipeline) is one instruction per segment time or machine cycle.
      - This translates into a speedup of k where k is the number of segments.
      - This is the ideal. In practice the speedup is less than k since the minimum time required by each stage will not be exactly balanced.
      - We must slow everything down to match the slowest stage.
      - In addition there is an overhead associated with each stage.
  - Pipelining is **almost** invisible to the programmer.

Additions and changes are required to pipeline the processor.
1. The PC must be capable of being incremented on each clock cycle. This requires an incrementer circuit that is independent of the ALU.
PC ← PC
PC ← PC + 4
2. Latches are needed to hold values that are required later in pipeline. IR, ALU, next PC, etc.

HAZARDS

- Hazards prevent pipeline from achieving ideal performance.
  - Much of this chapter will deal with eliminating hazards.
  - There are three types of hazards.

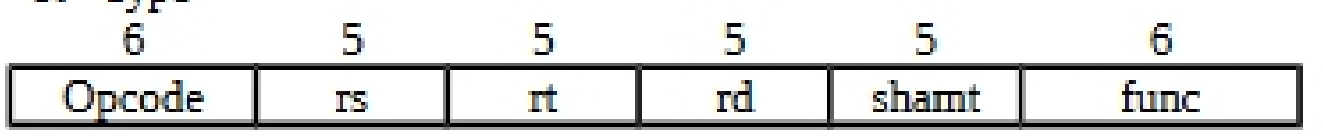
1. Structural Hazard - Resource conflicts.
   Memory is needed to load data and fetch instruction at the same time.
   Typically processor will have separate cache memories for instruction and data.
   with 95% or better hit rate.
2. Data Hazard - Instruction depends on result of previous instructions that has not been stored.
   ADD R1,R2,R3
   SUB R4,R5,R1
3. Control Hazard - Branches and other instructions that change the PC.
   ADD R1,R2,R3
   BEQZ R1,NEW
   ADD R4,R5,R6

Hazards cause pipeline to stall.

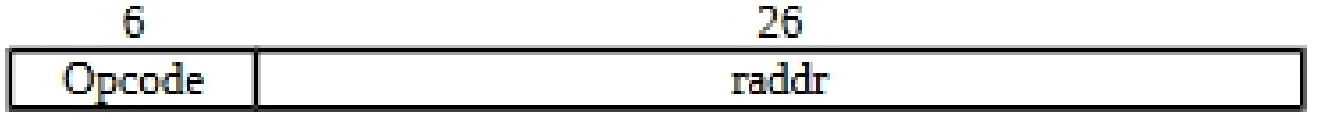
Version 1.1

I - Type instruction

R - Type



J - Type



DEFINITIONS

op = op(5..0) = IR(31..26) Rs = Rs(4..0) = IR(25..21) Rt = Rt(4..0) = IR(20..16)
Rd = Rd(4..0) = IR(15..11) n = n(15..0) = IR(15..0) ofst = ofst(15..0) = IR(15..0)
shamt=shmat(4..0)=IR(10..6) func = func(5..0)=IR(5..0) addr = addr(25..0) = IR(25..0)

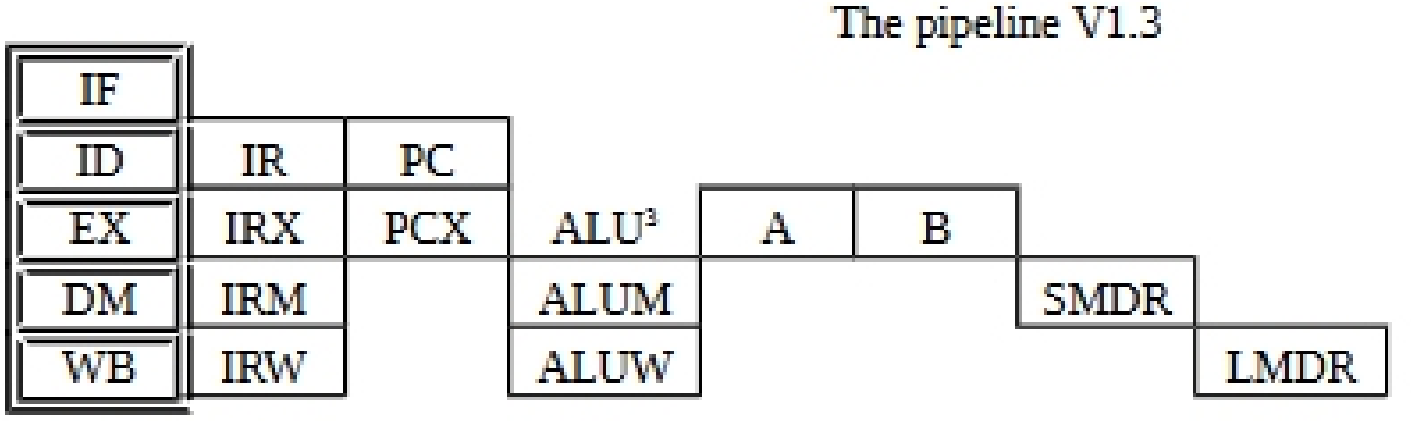
When referring to bits in the IR, Rs, Rt, and Rd to represent the bits that address GPR'r. Otherwise rs, rt, and rd will be used to represent the contents of particular GPR's. Specifically: rs = GPR(IR(25..21)), rt = GPR(IR(20..26)), and rd = GPR(IR(15..11). The proper interpretation should be obvious form the context.

The MIPS (KIPS) Instruction Set

lw	rt,ofst(rs)	rt ¬ M(rs + ±ofst)
sw	rt,ofst(rs)	M(rs + ±ofst) ¬ rt
beq	rs,rt,ofst	if rs=rt then PC¬PC +±ofst‖00
		else PC ¬ PC+4
add	rd,rs,rt	rd ¬ rs + rt
sub	rd,rs,rt	rd ¬ rs - rt
and	rd,rs,rt	rd ¬ rs & rt
or	rd,rs,rt	rd ¬ rs | rt
slt	rd,rs,rt	If rs < rt[2] then rd ¬1 else rd ¬0
addi	rt,rs,n	rt □ rs + ±n
andi	rt,rs,n	rt □ rs & ±n
j	addr	PC ¬ PC(31..28)‖addr‖00
jr	rs	PC □ rs

[1] These 5 bits specify which register is rs.
[2] rs and rt are treated as signed integers.

The pipeline V1.3



Pipelined Micro Operations Version 1.0

IR ← IR	IR ← IM(PC)	
IRX ← IRX[4]	IRX ← IR	
IRM ← IRM	IRM ← IRX	
IRW ← IRW	IRW ← IRM	
ALUM ← ALUM	ALUM ← SA op SB[5]	
ALUW ← ALUW	ALUW ← ALUM	
PC ← PC	PC ← PC + 4	PC ← PCX + ↓IRX.n‖00
PCX ← PCX	PCX ← PC	PC ← PCX(31..28)‖IRX.addr‖00
A ← A	A ← rs	
B ← B	B ← rt	
Ri ← Ri[6]	Ri ← ALUW[7]	Ri ← LMDR
SMDR ← SMDR	SMDR ← B	Data memory write
LMDR ← LMDR	LMDR ← DM(ALUW)	DM(ALUW) ← SMDR
ZERO[8] ← 0	ZERO ← 1	
SA := 0	SB := 0	
SA := 1	SB := 1	
SA := 4	SB := 4	
SA := A	SB := B	
	SB := ↓IRX.n	
SA := ALUM	SB := ALUM	
SA := ALUW	SB := ALUW	

[3] This is the combinational logic associated with the ALU. The output of the combinational ALU is normally transferred into the register ALUM on a clock transition.
[4] When pipeline stalls
[5] op ∈ {+, - , &, …}
[6] i ∈ {0, 1, …,31}
[7] "↓" means that the transfer takes place on the clock edge before the transfer caused by "←". This will be explained later.
[8] ZERO is a one bit register that contains the zero output of the ALU.