Andrew Hutzel (915841776)

1)

a. In what fraction of all cycles is the data memory used?

   lw + sw = 25 + 10 = 35%

b. In what fraction of all cycles is the input of the sign-extend circuit needed? What is the circuit doing in cycles in which the input is not needed?

   Addi + beq + lw + sw = 20 + 25 + 25 + 10 = 80%

2.

A. Branch Prediction
    - Addressed control huzzards
        - It achieves this by presuming the outcome of a branch instruction and then executing the instructions on one side of the branch to perpetually move the pipeline forward. Predictions can be made in hardware or software by the compiler.

B. Instruction scheduling
    - It addresses structural huzzards and data huzzards.
        - A compiler optimization used to improve instruction-level parallelism, which improves performance on machines with instruction pipelines. Without altering the code this attempts to: avoid pipeline stalls by re-arranging instruction order and avoid illegal or ambiguous operations.

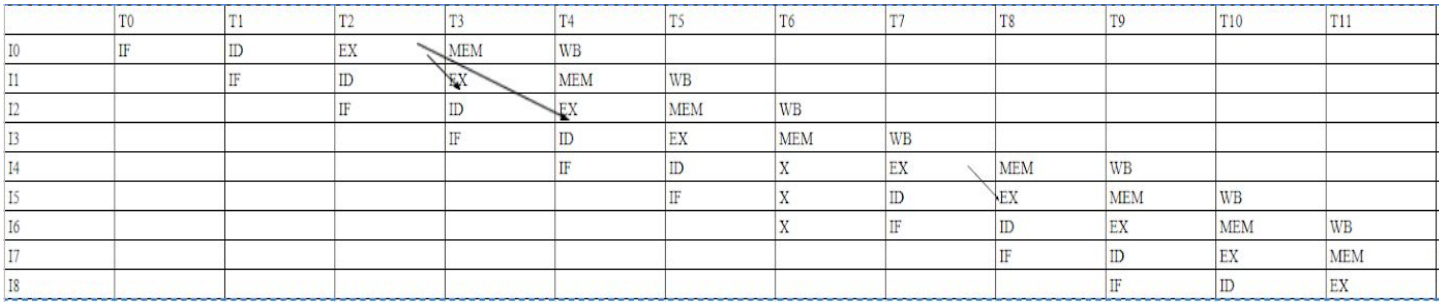
C. Delay slots
    - Addresses control huzzards
        - Helps avoid stalls that would happen due to the branch target identification during the decode stage (or after that) by scheduling the execution of some other instruction which has to be executed without concern for the branch condition. Symbiosis between the hardware and the given compiler is required.

D. Increasing availability of functional units
    - Helps avoid structural hazards. Has the ability to run multiple instructions of the same type simultaneously if we have replicated functional units.

E. Caches
    - Addressed data huzzards. Caches help reduce memory latency and reduce load-use latency which helps reduce the stall duration and improve execution time.

3.

Answer: **Execution time of code is 13 cycles.**



	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
I0	IF	ID	EX	MEM	WB							
I1		IF	ID	EX	MEM	WB						
I2			IF	ID	EX	MEM	WB					
I3				IF	ID	EX	MEM	WB				
I4					IF	ID	X	EX	MEM	WB		
I5						IF	X	ID	EX	MEM	WB	
I6							X	IF	ID	EX	MEM	WB
I7									IF	ID	EX	MEM
I8										IF	ID	EX

Note: Feel free to add more cycles (T9, T10, etc) as much as you may require. Here, I0, I1, etc stand for the instructions and T0, T1, etc stand for the cycle times.

4.

RAW dependence		WAR dependence		WAW dependence	
From Instr	To Instr	From Instr	To Instr	From Instr	To Instr
I0	I1	I0	I1	I0	I3
I0	I2	I1	I3	I1	I4
I1	I2	I2	I4		
I3	I5	I3	I4		
I2	I3	I4	I5		
I1	I3	I5	I6		
I2	I4				
I3	I5				
I5	I6				

5.

A. What is the clock cycle time in a pipelined and non-pipelined processor?

   Pipelined:
   The clock cycle time has to accommodate the slowest operation. So in this case, the slowest operation is ID, which means the cycle time is 350 ps.

   Non-pipelined:
   The clock cycle time is the sum of all the stages.
   250 ps + 350 ps + 150 ps + 300 ps + 200 ps = 1250 ps

B. What is the total latency of an LW instruction in a pipelined processor and a non-pipelined processor?

   Pipelined:
   Total latency = Cycles × Clock Cycle time
   5 × 350 = 1750 ps

   Non-pipelined:
   Total latency = sum of all the stages
   250 + 350 + 150 + 300 + 200 = 1250 ps

C. If we can split one stage of the pipelined data-path into two new stages, each with half latency of the original stage, which stage would you split and what is the new cycle time of the processor?

   I would split Instruction Decode since it has the largest latency which is what ultimately defines the total latency. The new cycle time would be defined by the next longest stage: 300ps MEM.

D. Assuming that there are no stalls or hazards, what is the utilization of the data memory?

   LW + SW = 35%

E. Assuming that there are no stalls or hazards, what is the utilization of the write register port of the "Registers" unit?

LW + ALU = 65%

F. What are the values of all inputs for the “Registers” unit?

Not enough information is given, needs clarification or a more concise wording of this here question.

Register Number	Conventional Name	Usage
$0	$zero	Hard-wired to 0
$1	$at	Reserved for pseudo-instructions
$2 - $3	$v0, $v1	Return values from functions
$4 - $7	$a0 - $a3	Arguments to functions - not preserved by subprograms
$8 - $15	$t0 - $t7	Temporary data, not preserved by subprograms
$16 - $23	$s0 - $s7	Saved registers, preserved by subprograms
$24 - $25	$t8 - $t9	More temporary registers, not preserved by subprograms
$26 - $27	$k0 - $k1	Reserved for kernel. Do not use.
$28	$gp	Global Area Pointer (base of global data segment)
$29	$sp	Stack Pointer
$30	$fp	Frame Pointer
$31	$ra	Return Address
$f0 - $f3	-	Floating point return values
$f4 - $f10	-	Temporary registers, not preserved by subprograms
$f12 - $f14	-	First two arguments to subprograms, not preserved by subprograms
$f16 - $f18	-	More temporary registers, not preserved by subprograms
$f20 - $f31	-	Saved registers, preserved by subprograms