Instructions
Instruction Level Parallelism
Pipelining

CPU

- Circuitry which implements a set of instructions
- Instructions have a binary number associated with them
  - But also a short string --- assembler instruction
- Registers to store data and instructions
  - Registers are numbered
- Clock that beats at a rate corresponding to chip clock rate
  - instructions happen synchronous to clock beats
- Program Counter

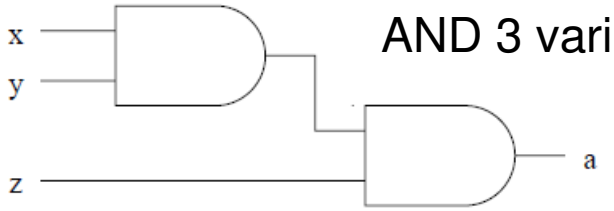
Programs/Program Counters

- CPU executes a sequence of instructions stored in memory.
- Today CPU's are usually load-and-store machines.
- They have many registers on which operations are performed.
- Load and store instructions are used to move data to and from memory.
- Instructions have addressing modes that aid in computing the address of a memory reference.
- The execution of the CPU is synchronized by a clock.
  - time between two ticks of a clock is called a cycle,
  - speed of the CPU is measured in cycles per second, aka hertz.
- Execution of an instruction usually takes several cycles,
- Address of the current instruction is contained in a special register called program counter (PC).
- After each instruction PC is incremented by instruction length, unless the result of the instruction changes the PC
- Jump and branch instructions override the natural sequence.

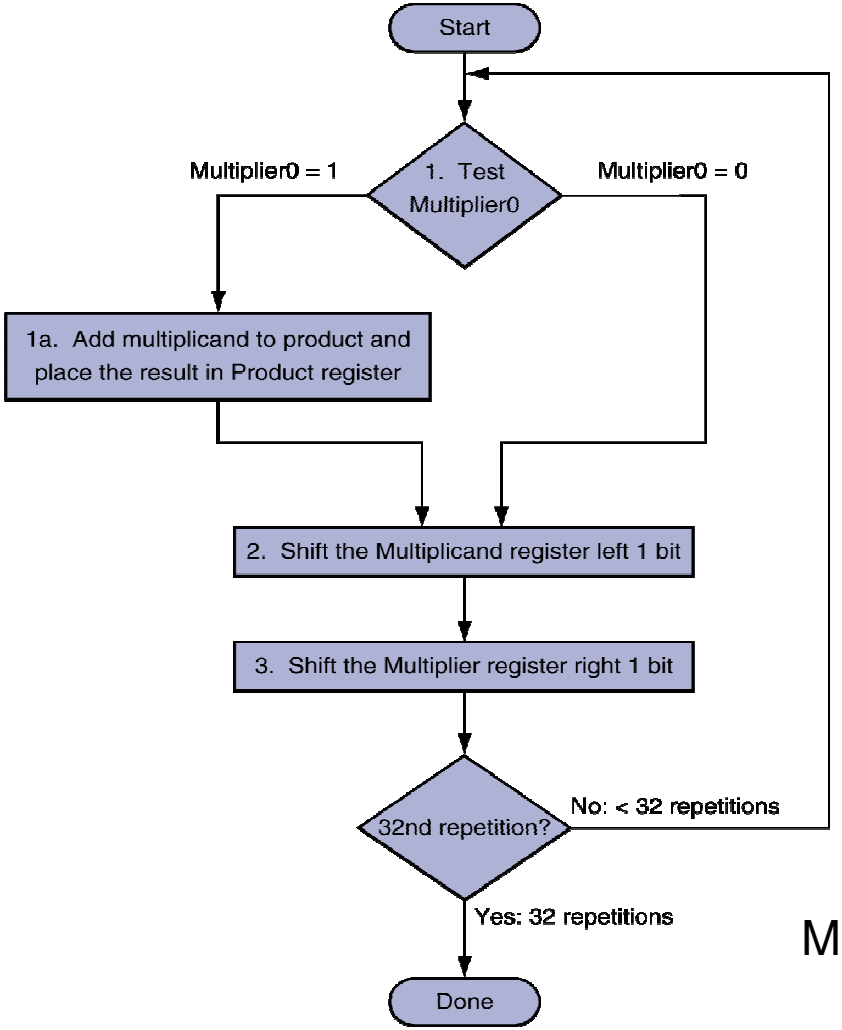
- A branch changes the PC to one of two values depending on whether a certain condition is satised.
- Branches are used to implement if statements and loops.
- Another instruction is a jump and save PC, which saves the value of the PC before changing it
  - program knows how to get back to where it was.
  - used to implement subprogram calls

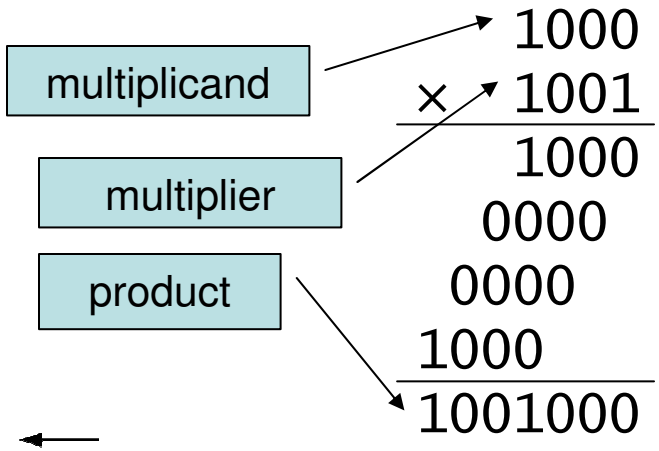
Instructions and hardware

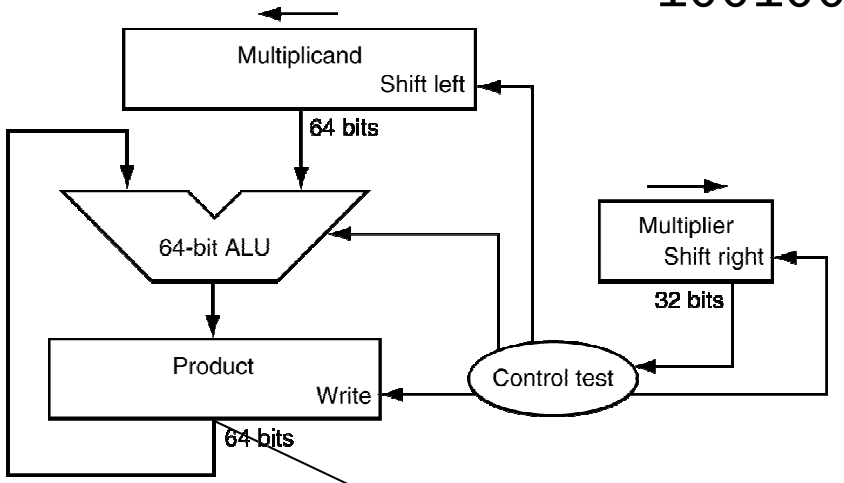
comparator
AND gate



AND 3 variables







Multiplication of integers

Initially 0

Instructions

- Bottom line: somehow instructions are mapped to hardware
- At beginning of a cycle instruction in special register called instruction register (IR).
- At clock tick, data flows from the IR and the registers through logic circuits.
  - logic circuits produced desired results which pile up at the registers.
- At cycle end, results are in registers and CPU proceeds to next instruction.
- Instruction usually takes several clock cycles to execute.
  - one cycle instruction fetched from memory to IR.
  - next, instruction decoded to send control signals to logic circuits.
  - Next cycle may be required to compute the results of the instruction,
  - another to store result in appropriate registers.
- CPU must have internal or working registers to store intermediate results.
- These are invisible to programmer. Cannot be directly written or read.
- Different CPUs may require different number of instructions to execute same operation
  - highlights limitations of the clock speed alone as a measure of performance.
- Performance depends on nature of the instructions computer can execute, how much their execution can be overlapped or pipelined, and how quickly items can be transferred between the CPU and memory.

INTEL

EM64

Architecture	NetBurst		Earlier Pentium Family Processors	
Port	µops	Throughput / Latency	µops	Throughput / Latency
0	Integer ALU†	0.5 / 0.5	Integer ALU (add, sub etc.)	1 / 1
	Integer store data	1 / NA	Integer shift	1 / 1
	FPU / MMX / SSE / SSE2 move	1 / 6	Integer multiply	1 / 4
	FPU exch	1 / 0	LEA instruction	1 / 1
	FPU / MMX / SSE / SSE2 store data	16-128 bits: 1 / NA	FPU add/sub	1 / 3
			FPU mul	2 / 5
			FPU div	37 / 38
			MMX ALU	1 / 1
			MMX mul	1 / 3
			SSE mul	2 / 5
			SSE div, sqrt	36-58 / 36-58
			SSE move	1 / 1
1	Integer ALU†	0.5 / 0.5	Integer ALU (add, sub etc.)	1 / 1
	Branches jcc, call, ret	0.5 / NA, 1 / 5, 1 / 8	MMX ALU	1 / 1
	Integer shift, rotate by 1	1 / 4	MMX shift, rotate, shuffle, pack, unpack	1 / 2
	Adc, sbb instructions	Reg, reg: 3 / 8, reg, imm: 2 / 6	SSE add / sub	2 / 4
	Inc, dec instructions	1 / 1	SSE shuffle	1-4 / 1-2
	Complex integer	NA	SSE reciprocal, reciprocal sqrt	2 / 2
	FPU / SSE / SSE2 add, sub	FPU: 1 / 5, SSE / SSE2: 2 / 4	SSE move	1 / 1
	Integer / MMX / FPU / SSE / SSE2 mul	Int: 3-5 / 14-18, MMX: 1 / 8, FPU: 2 / 7, SSE / SSE2: 2 / 6		
	Integer / FPU / SSE / SSE2 div, sqrt	Int: 23 / 56-70, FPU: 23-43 / 23-43, SSE: 32 / 32, SSE2: 62 / 62		
	FPU misc	NA		
	MMX / SSE / SSE2 ALU	MMX: 1 / 2, SSE / SSE 2: 2 / 2		
	MMX / SSE / SSE2 shift, rotate, shuffle, pack, unpack	MMX: 1 / 2, SSE / SSE 2: 2 / 2		
	MMX misc, SSE / SSE2 reciprocal	NA, 4 / 6		
2	Load data / Prefetch	8-128 bits: 1 / 2	Load data	8-64 bits: 1 / 3, 128 bits: 2 / 4
3	Store address generation	1 / 1	Store address generation	1 / 1
4	N/A	&nbs p;	Store data	8-64 bits: 1 / NA, 128 bits: 2 / NA

Instructions and times on a GPU

- 4 clock cycles for:
  - single-precision floating-point add, multiply, and multiply-add,
  - integer add,
  - bitwise operations, compare, min, max, type conversion instruction;
- 16 clock cycles for reciprocal, reciprocal square root, **__logf(x)**
- 32-bit integer multiplication takes 16 clock cycles,
- **__mul24** and **__umul24** provide signed and unsigned 24-bit integer multiplication in 4 clock cycles.
- Single-precision floating-point square root is implemented as a reciprocal square root followed by a reciprocal instead of a reciprocal square root followed by a multiplication, It takes 32 clock cycles for a warp.
- Single-precision floating-point division takes 36 clock cycles, but another instruction **__fdividef(x,y)** provides a faster version at 20 clock cycles
- **__sinf(x)**, **__cosf(x)**, **__expf(x)** take 32 clock cycles.
- **sinf(x)**, **cosf(x)**,**tanf(x)**, **sincosf(x)** are much more expensive and even more so (i.e. about an order of magnitude slower) if the absolute value of **x** is greater than 48039 (see **math_functions.h** for more details). Moreover, in this case, the argument reduction code uses local memory, which can affect performance even more because of local memory high latency and bandwidth.
- Special instructions for doing floating point multiply and add as one instruction

MIPS instruction set

- The MIPS64 computer (here shortened to MIPS) is a embedded computer that is a computer designed to be incorporated into devices, like cell phones and hand calculators, that require computer assistance.
- The 64 refers to the fact that it operates with 64 bit words.
- The MIPS has 32 general purpose registers named R0, R1, . . . , R31, each containing 64 bits.
- The register R0 always contains zero. The MIPS also has 32 double-precision floating-point registers F0, F1, . . . , F31, which can also be used for single-precision numbers.
- Also special internal and instruction registers

THE MIPS PIPELINE

1. Instruction fetch (IF): The instruction is fetched from memory and placed in the instruction register (IR).

2. Instruction decode (ID): The bits of the instruction are decoded into control signals. Operands are moved from registers or immediate fields to working registers. For branch instructions, the branch condition is tested and the branch address computed.

3. Execution (EX): The instruction is executed. Specifically, if the instruction is an arithmetic or logical operation, its results are computed. If it is a load-store instruction, the address is computed. All this is done by an elaborate logic circuit called the arithmetic-logical unit (ALU).

4. Memory read/write (ME): If the instruction is a load-store, the memory is read or written.

5. Write back (WB): The results of the operation are written to the destination register.

STRUCTURAL HAZARDS

cycle	IF	ID	EX	ME	WB
1	I1				
2	I2	I1			
3	I3	I2	I1		
4	I4	I3	I2	I1	
5	I5	I4	I3	I2	I1
6	I6	I5	I4	I3	I2
.	.	.	.	.	.

suppose I2 is a load. Then we have the following execution sequence

cycle	IF	ID	EX	ME	WB
1	I1				
2	LD	I1			
3	I3	LD	I1		
4	I4	I3	LD	I1	
5	–	I4	I3	LD	I1
6	I5	–	I4	I3	LD
6	I6	I5	–	I4	I3
.	.	.	.	.	.