A processor has a set of registers, an execution unit, and a control unit. The execution unit executes processor operations, while the processor activation control unit generates control signals.In each processor, there are two registers directly connected to the outside processor system bus. Those registers are MAR (memory address register) and memory buffer register(MBR). MAR is directly connected to the system bus address lines and MBR directly to the system bus data lines.

Program execution is nothing but a sequence of instruction cycle execution. Each instruction has four sub-instruction cycles: fetch, indirect operand fetch, execution, and interrupt. Again, each sub instruction cycle has micro-operations.

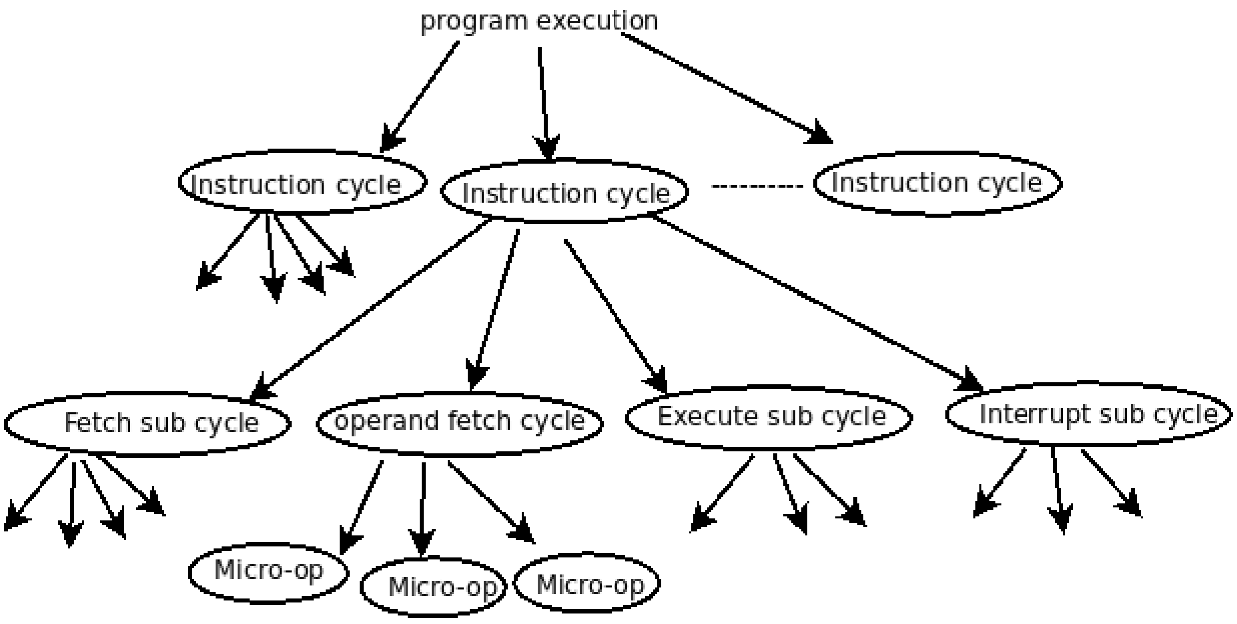

Figure 1 sub instruction cycles with micro-operations

Fetch Sub instruction cycle

In the fetch sub-instruction cycle**, instructions are** fetched from memory to processor registers. There are different micro-operations performed in different processor time units or CPU clock pulses.

T1: MAR <---PC
T2: MBR <--- Memory
    PC <--- (PC)+1
T3: IR <---MBR

The program counter (PC) register address is loaded into the memory address register (MAR) in the T1 time unit. The MAR register is directly connected with the system bus address lines. Based on this address, an instruction was loaded into the processor's MBR register in the T2 time unit, and the PC value was increased by one. Now the PC is pointing to the next instruction memory location. In T3, a time unit instruction is loaded into the instruction register(IR) from the memory buffer register(MBR).

Indirect sub instruction cycle

After the instruction decode stage, the processor knows which operation needs to be performed and how many operands are needed for this instruction execution. Based on instruction, the most significant bit value operand is fetched in direct or indirect addressing mode. Assume that an MSB value of 0 indicates that the instruction operand is being fetched from memory in direct addressing mode. If the MSB value is 1, it indicates that the instruction operand is being fetched from memory in indirect addressing mode. In this indirect sub-instruction cycle, the instruction operand is fetched from memory in an indirect addressing mode.

T1: MAR <--- (IR(address))
T2: MBR <--- Memory
T3: IR(address) <--- (MBR(address))

In T1, the operand address of the time unit instruction is loaded into the MAR register, which is directly connected to the outside processor system bus. In the T2 time unit, based on the instruction address operand loaded from memory to the MBR register, In T3, the time unit operand is loaded from MBR to the instruction operand field in IR. Above, three micro-operations are performed in different processor time units.

Execute sub instruction cycle

In this execute sub instruction cycle, some of the micro-operations for ADD R1, X instruction execution are performed. To execute this instruction requires two operands; one operand in the R1 register and the second operand needs to be fetched from memory.

T1: MAR <--- (IR(address))
T2: MBR <--- memory
T3: R1 <--- (R1) + (MBR)

The instruction operand address is loaded into the MAR register in T1 time unit, and the operand is loaded from memory to the MBR register in T2 time unit based on this address. R1 register value and MBR register value are added in T3 time unit and the result is stored in R1 register.

Interrupt sub instruction cycle

In this cycle, after instruction execution, the processor checks if any interrupt occurred or not. If an interrupt occurs, then these micro-operations are performed.

T1: MBR <--- (PC)
T2: MAR <--- Save_address
PC <--- Routine_address
T3: Memory <--- (MBR)

The next instruction address is saved back into memory for future execution in T1 time unit programme counter (PC) contents. In the T2 time unit, there are two micro-operations. In the first micro-operation, a saved address is transferred into the memory address register (MAR), and in the second micro-operation, an interrupt routine address is assigned to the program counter(PC). In T3, the previous instruction address(PC) in MBR is transferred into memory.

Control Unit Design

The control unit generates control signals to activate various components in the processor, like registers, internal bus, ALU, and paths between various components in the CPU. Condition codes, instruction register operation codes, external inputs, and the clock are all inputs to the control unit.

Instruction Register: At present, the executed instruction is loaded into the IR register from the memory buffer register (MBR). After the instruction decoder knows which operation has to be performed, these operations bits are input to the control unit.

Condition codes (flags) : Condition flag bits are inputs for the control unit. Condition codes give the status of the previous instruction operation. For example, in Increment and Skipp, if zero(ISZ) instruction effective address is incremented by one and compared with zero, if equal means Zero Flag (ZF) flag set to 1, the processor skips the next instruction execution.

External Inputs: Control signals are provided from the system bus, like waiting until memory function is completed (WMFC). Control signals are generated from the main memory. From input devices, interrupt request signals are also generated to control the unit.

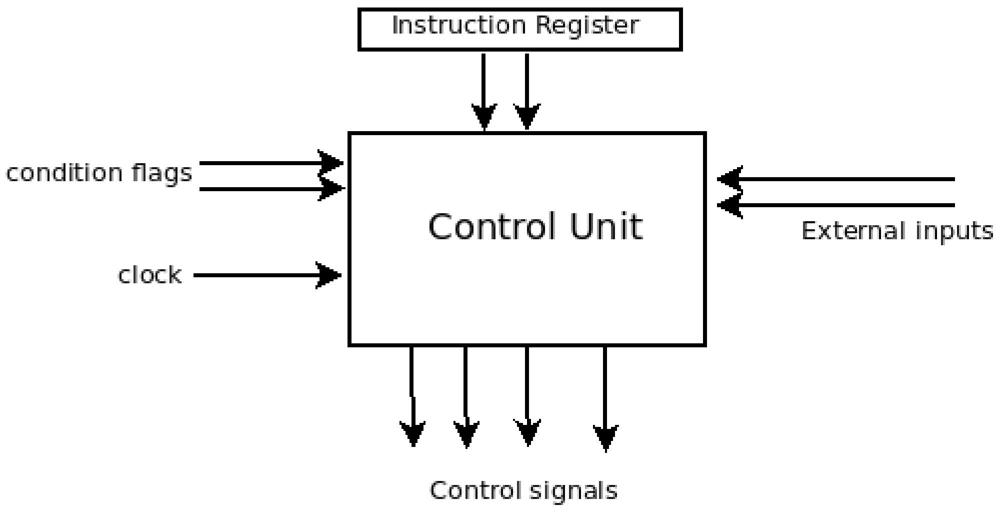

figure 2 Basic diaram of control unit

Clock: In a processor clock cycle or clock pulse, one micro-operation or a set of simultaneous micro-operations is performed.

Hardwired Control Unit

The processor has to generate control signals in the proper sequence for executing instructions. For hardwired control design, hardware components like AND gates, OR gates, encoders, and decoders are used. A counter is used to keep track of the control steps. Inputs for the hardwired control unit are the clock, instruction register, condition codes, and external inputs like WMFC and interrupts.

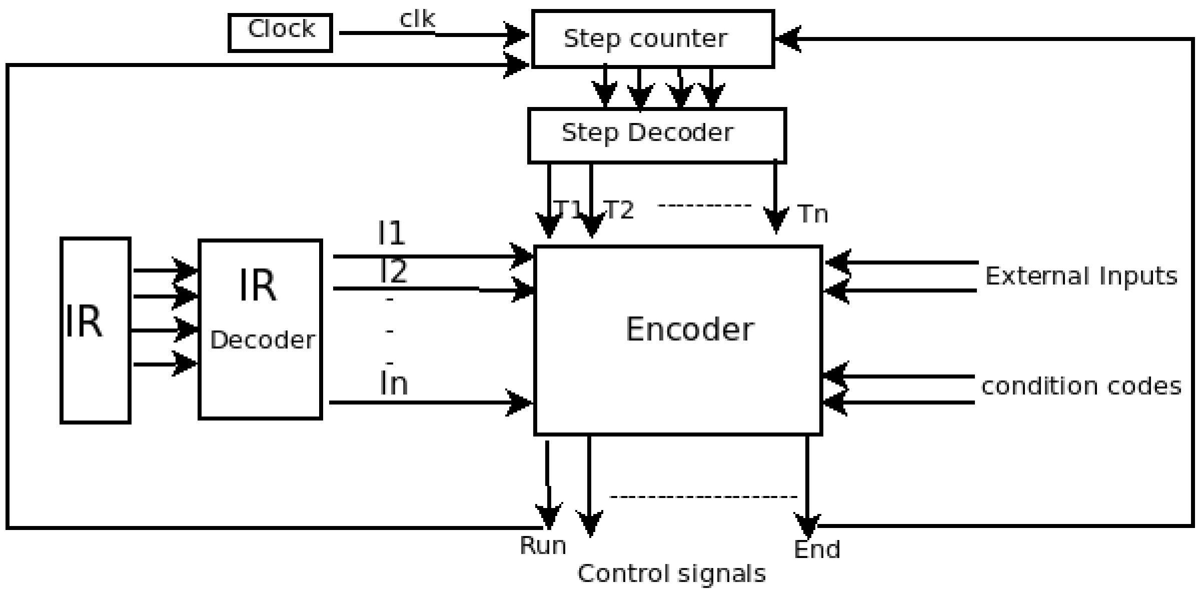

Figure 3 Hardwired control unit

Each step in this sequence is completed in one clock cycle. The step decoder provides a separate signal line for each time unit in the control sequence. instruction register opcode bit inputs for the IR decoder. The output of the instruction decoder consists of a separate line for each machine instruction. For any instruction loaded in the IR register, one of the output lines from I1

through In is set to 1 and the other lines are set to 0, which means not active. The encoder combines all the input signals and generates separate control signals like Yin, PC Out, ADD, Run, End, etc.
Zin = T1+T6.ADD+T4.BR+-----

The End control signal starts a new instruction fetch cycle by resetting the control step counter to its starting value. When the RUN control is called and set to 1, it means the step counter is incremented by one at the end of every clock cycle. When RUN reaches 0, the step counter stops counting.

- Hardwired control units are difficult to modify because they have an array of logic (AND,OR) gates with a lot of wiring.
- A hardwired control unit uses reduced instruction set computer (RISC) architecture.
- Hardwired control units are more costly as compared with microprogrammed control units.
- Hardwired CU is able to generate control signals for limited instructions.

Microprogrammed Control Unit

The microprogrammed control unit also generates a sequence of control signals for instruction execution. It operates on the basis of a complex instruction set computer (CISC). The programme which is used to generate control signals is called "Microprogrammed". This microprogrammed is placed on the processor chip. It is a special type of memory, also known as control memory.

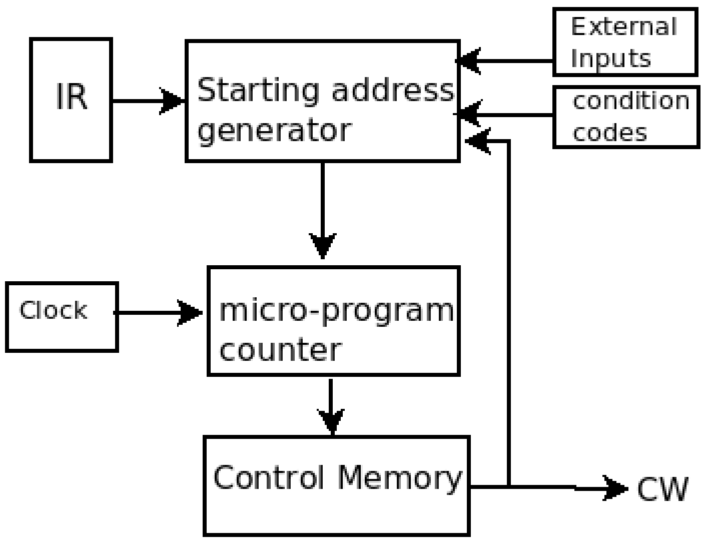

Figure 4 Microprogrammed control unit

Each micro-program contains a set of microinstructions. Each microinstruction or control word contains different bit patterns. Contorl words contain binary 1’s and 0’s and, based on control word bits, there are different control signals generated. A micro-routine is nothing but a sequence of microinstuctions.

- The instruction register, which contains the currently executed instruction, external inputs from the system bus such as WMFC, interrupts, and clock are all inputs to the microprogrammed control unit.
- The Statring address generator fetches instructions from the instruction register and instructions are decoded. After that, the first micro-routine or instruction starting address is loaded into the micro-program counter .
- Based on this micro-program counter address control word generated from control memory, with this control signal, the micro-routine starts execution.

- The micro-program counter increments automatically with the clock so that it can read the sequence of control words of a micro-routine.
- If all micro-routines are successfully completed, then the end bit is set to 1, which means the microprogrammed control unit can start the next instruction execution.

Advantages and Disadvantages

- It is less costly as compared to a hardwired control unit because it is implemented with a program.
- It is easy to modify compared with the hardwired control unit. If we change instructions, it can be modified, but the hardwired control unit is difficult to modify because it has an array of logic gates and a lot of wiring.
- The microprogrammed control unit is slower compared to hardwired control because it generates control signals for many instructions.
- Hardwired control uses the RISC architecture, and microprogrammed control uses the CISC architecture.

A microprogrammed horizontal control unit

- In a horizontal microprogrammed control unit, n bits in a control word will generate n control signals.
- Some bits for external inputs, some bits for functional codes like conditional jump, some bits for condition codes (flags), and some bits for the next instruction address.
- In this control unit, a control word has a greater number of bits (longer).

The vertical microprogrammed control unit

- In this control unit for n bits only, logn control signals in the control word.
- Suppose there are 64 bits for functional codes. Instead of 64 control signals, there are only 6 control signals for all 64 bit functional codes.
- A control word has a lower number of bits (shorter).

Instruction Pipelining

In instruction pipeling, multiple instructions are executed in a single processor clock cycle. But in non-pipeline architectures, instructions are executed in sequence, one after another. In the below figure, there are 3 instructions with 4 stages (fetch, decode, execute, and write back). Each state is executed in one processor clock cycle. Executing 3 instructions in a pipeline requires 6 processor clock cycles. In non-pipeline mode, executing 3 instructions with 4 stages requires 12 processor clock cycles.

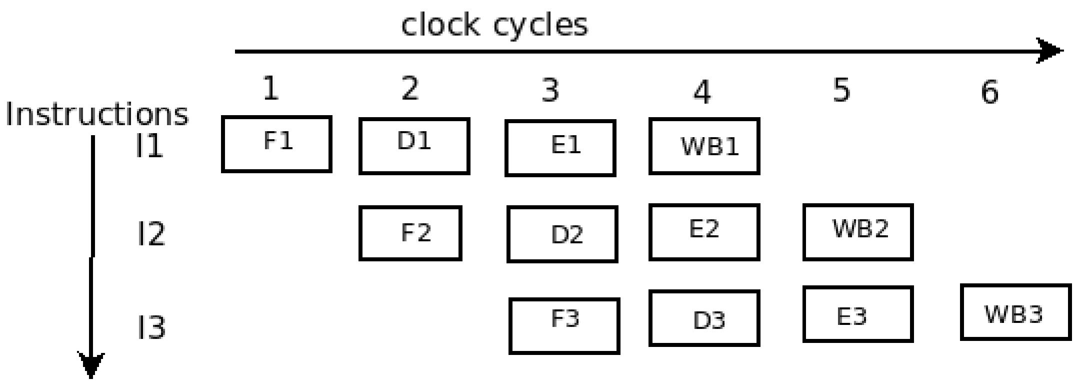

Figure 5 Instruction execution in pipelining

Suppose there are "N" instructions with K-stages and the maximum time delay from one instruction stage to another is T time units. The difference in time required to execute an instruction with and without a pipeline is
(K+N-1)T is the time required to execute n instructions in a pipeline.

K-number of stages

N-total number of instructions

T-time delay from one state to another

NKT is the time required to execute instructions in a nonpipeline.

Speed up factor =NK/(K+N-1)

There are 3 types of pipeline hazards that occur in pipeline instruction executions.

Resource or Structural Hazards

In this hazard, if multiple instructions are in the pipeline, Assume that main memory has a single port through which this processor has to perform read or write operations one at a time. Suppose two or more instructions need the same resource (main memory) in a single processor clock cycle. In this case, a resource hazard occurs.

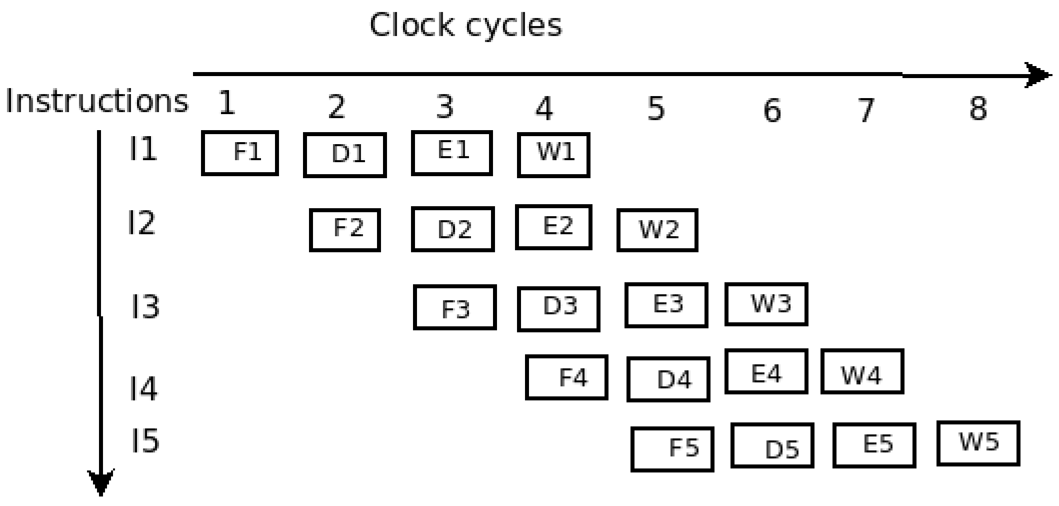

Figure 6 Instruction resource hazards

In Figure 6, instruction I1 writes operations and I4 fetches operations in the processor's 4th clock cycle. Instruction I2 write operations and I5 fetch operations in the processor's 5th clock cycle. In the 4th clock cycle and 5th clock cycle, main memory clashes occur. This is a problem in the pipeline if more than two instructions perform different operations on the same resource (main memory).

Resource Hazard Solution

To avoid the above problem, after completion of the first 3 instructions, the next instructions must be executed in a sequence.

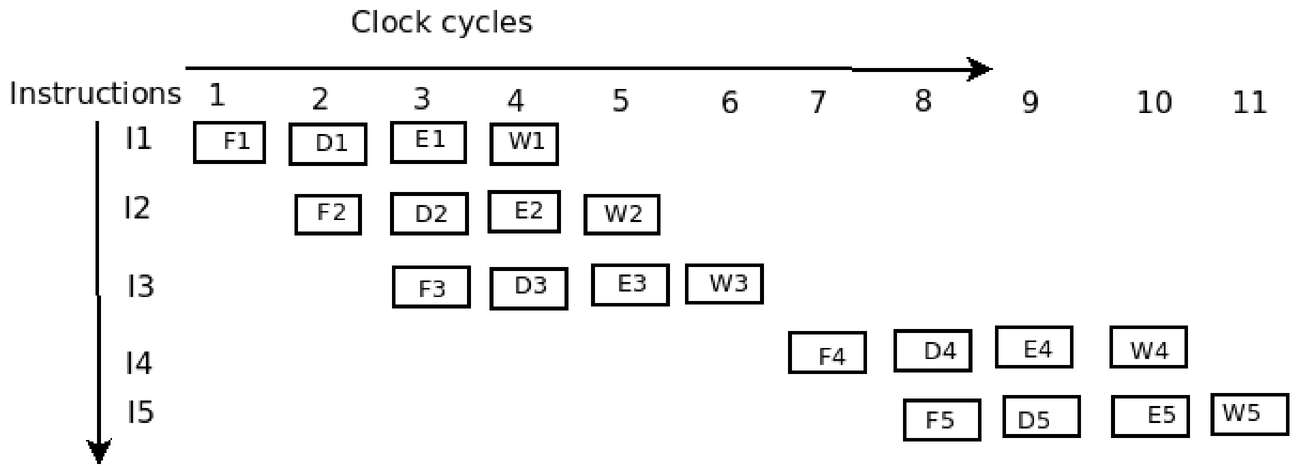

Figure 7 resource hazard solutions

Data Hazards

Data hazards occur if two instructions are in the pipeline and are executed parallly. In the event that data hazards occur, the second instruction execution is dependent on the first instruction result. If instructions are executed in a sequence, then there is no problem and the program will give correct results. If both instructions are executed in the pipeline and the programme is dependent on the previous result, the programme will produce incorrect results. In the below example, data hazard occurs.

Example:

ADD EAX, EBX
SUB ECX, EAX

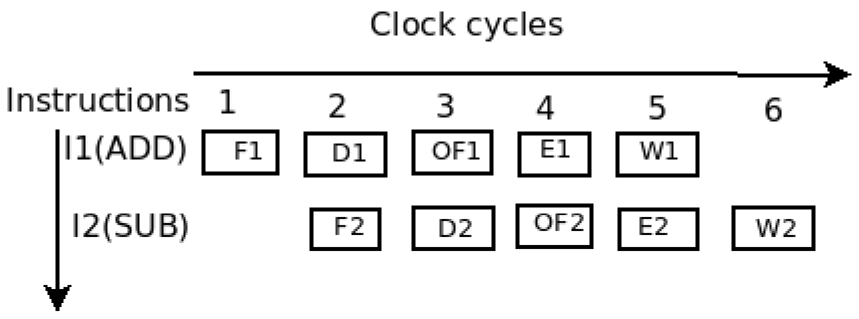

Figure 8 Instruction pipeline Data Hazard

In figure 8, instruction I1 successfully executes after 5 clock cycles, but second instruction I2 (sub) fetches no modified operands in the 4th clock cycle. In this situation, the program will give an incorrect result.

Data hazard solution

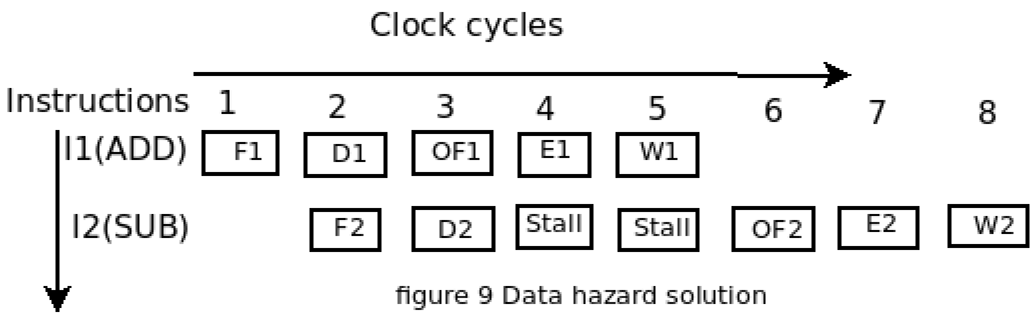

figure 9 Data hazard solution

Figure 9 is a solution for data hazards. During execution of instruction I1 for I2, the operand is not fetched from the previous instruction. After completion of instruction I1, instruction I2 fetched the operand in the 6th clock cycle. So, in clock cycles 4 and 5, instruction execution is idle. This is called pipeline stall.

Control Hazards

Control hazards occur for branch instructions. The processor will not know which instructions are branch instructions until after the decoding of instructions. Suppose there are 3 instructions(I1, I2 & I3) in the pipeline. Assume Instruction I2 is a branch instruction, but the processor knows after the decode stage of the instruction. In PC branch address loaded, the processor jumps to the branch instruction and starts execution.

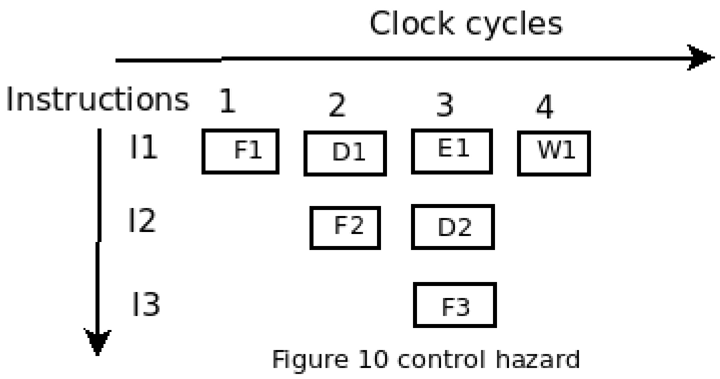

Figure 10 control hazard

In Figure 10, instuctions I1, I2 and I3 in the pipeline. Instruction I2 is decoded in the 3rd clock cycle and the processor knows it is a branch instruction, but during the decoding of the I2 instruction, the I3 instruction is fetched from main memory.

Solution for hazard control

The instruction I3 fetch operation was completely terminated from the pipeline prior to loading the branch instruction address into PC. Instruction Im is a branch instruction in the below mentioned figure 11.

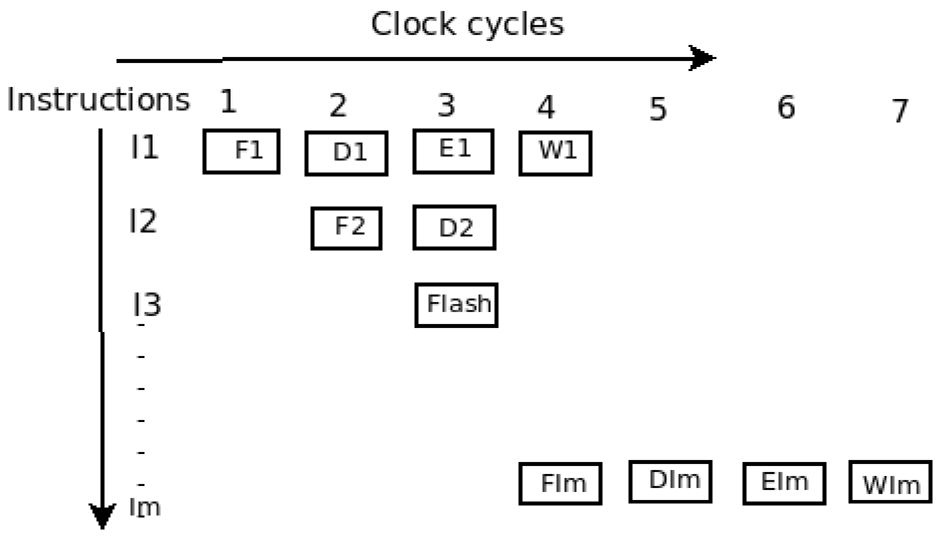

figure 11 control hazard solution