A quality product by
Brainheaters™

Bh.Notes: MP

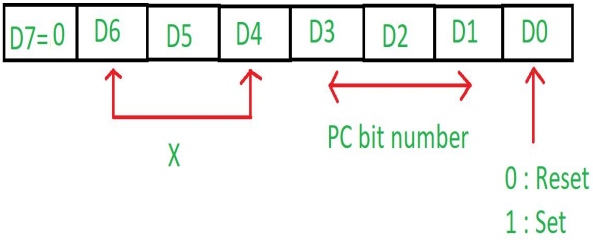
Computer Semester 4

A series of Important Concepts/Questions
highly recommended for MU Exam

'C' SCHEME - 2020-2021

Q1. Explain BSR mode of 8255 PPI.(P2-Appeared 3 Times) (5-10M)

ANS: Bit set reset (BSR) mode – This mode is used to set or reset the bits of port C only, and selected when the most significant bit (D7) in the control register is 0. Control Register is as follows:

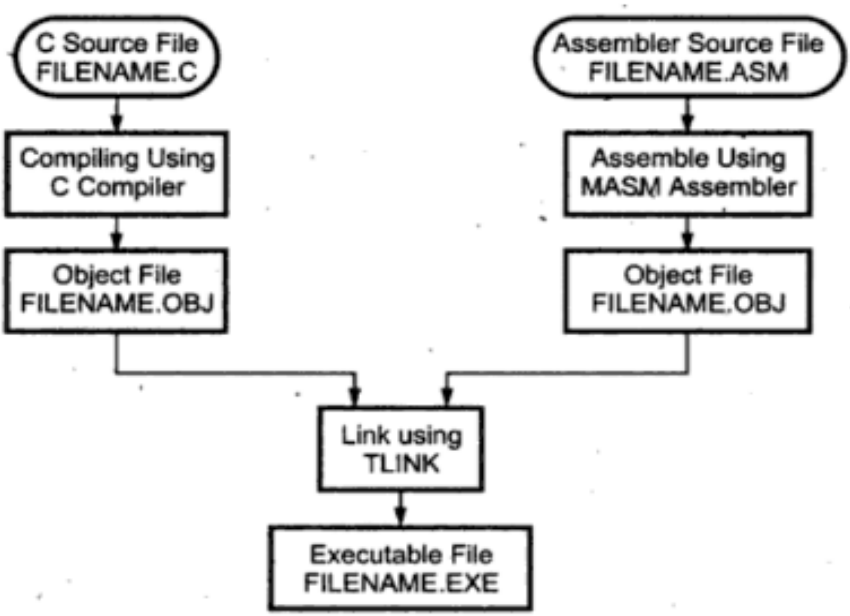


D3	D2	D1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

This mode affects only one bit of port C at a time because, as the user sets the bit, it remains set until and unless the user changes it. User needs to load the bit pattern in the control register to change the bit.

Q2. Write note on: Mixed Language Programming. (P4-Appeared 1 Times) (5-10M)

ANS: There are times when programs need to call programs written in other languages referred to as mixed language programming.



Compiler, assembly and link processes

- For example, when a particular subprogram is available in a language other than the language you are using, or when algorithms are described more naturally in a different language, you need to use more than one language.
- Mixed-language programming always involves a call to a function, procedure, or subroutine.
- Mixed-language calls involve calling functions in separate modules. Instead of compiling all source programs with the same compiler, different compilers or assemblers are used as per the language used in the programs.
- Microsoft C supports this mixed language programming. So it can combine assembly code routines in C as a separate language.
- C program calls assembly language routines that are separately assembled by-MASM (MASM Assembler).

- These assembled modules are linked with the compiled C modules to get an executable file. Fig shows the compile, assemble and link processes using C compiler, MASM assembler, and TUNIC.

Q3. Write note on: Code cache organization of Pentium. (P2-Appeared 3 Times) (5-10M)

ANS: Despite the potential advantages of a unified cache which is used in the 80486 processor, the Pentium microprocessor uses separate code and data caches.

- The reason is that the superscalar design and branch prediction demand more bandwidth than a unified cache.
- First, efficient branch prediction requires that the destination of a branch be accessed simultaneously with data references of previous instructions executing in the pipeline.
- Second, the parallel execution of data memory references requires simultaneous accesses for loads and stores.
- Third, in the context of the overall Pentium microprocessor design, handling self-modifying code for separate code and data caches is only marginally more complex than for a unified cache.
- The data and instruction caches on the Pentium processor are each 8 KB, two-way associative designs with 32 byte lines.
- Each cache has a dedicated translation lookaside Buffer (TLB) to translate linear addresses to physical addresses.
- The caches can be enabled or disabled by software or hardware. The Pentium microprocessor implements the data cache to support dual accesses by the U-pipe and V-pipe to provide additional bandwidth and simplify compiler instruction scheduling algorithms.
- The data cache is write back or write through configured on a line-by-line basis and follows the MESI protocol.

- The data cache tags are triple ported to support two data transfers and an inquiry cycle in the same clock.
- The code cache is an inherent write protected cache. The code cache tags of the Pentium processor are also triple ported to support snooping and split-line accesses.
- The data path, however, is single ported with eight way interleaving of 32-bit-wide banks. When a bank conflict occurs, the U-pipe assumes software or hardware.
- The Pentium microprocessor implements the data cache to support dual accesses by the U-pipe and V-pipe to provide additional bandwidth and simplify compiler instruction scheduling algorithms.
- The data cache is write back or write through configured on a line-by-line basis and follows the MESI protocol.
- The data cache tags are triple ported to support two data transfers and an inquiry cycle in the same clock.
- The code cache is an inherently written protected cache.
- The code cache tags of the Pentium processor are also triple ported to support snooping and split-line accesses.
- The data path, however, is single ported with eight way interleaving of 32-bit-wide banks.
- When a bank conflict occurs, the U-pipe assumes priority, and the V-pipe stalls for a clock cycle.
- The bank conflict logic also serves to eliminate data dependencies between parallel memory references to a single location.
- For memory references to double-precision floating-point data, the processor accesses consecutive banks in parallel, forming a single 64-bit path.

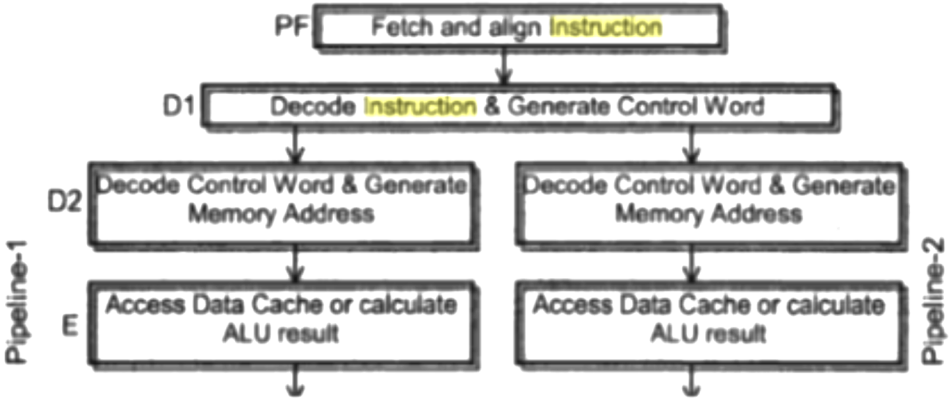
Translation lookaside buffers (TLB):

- Besides general-purpose caches, X86 processors include caches called Translation Lookaside Buffers (TLB) to speed up linear address translation.
- When a linear address is used for the first time, the corresponding physical address is computed through slow accesses to the page tables in RAM.
- The physical address is then stored in a TLB entry so that further references to the same linear address are quickly translated.
- When the CR_3 control register is modified, the hardware automatically invalidates all entries of the TLB.

Q4. Write the instruction issue algorithm used in Pentium.

(P2-Appeared 3 Times) (5-10M)

ANS: The Pentium has superscalar organizations. It enables 2-instructions to be executed in parallel.



(a) Superscalar execution

- Figure below (a) shows that the resources for address generation and ALU functions have been replicated in an independent integer pipeline, called U- and V-.
- The ????P in the PF and D1 stages can fetch and decode 2-simple instructions in parallel and issue them to the U- and V-pipelines.
- Additionally, the ????P for complex instructions in D1 can generate micro coded sequences that control both U- and V-pipelines.
- Several techniques are used to resolve dependencies between instructions that might be executed in parallel.
- Most of the logic is contained in the instruction issue algorithm as indicated in Figure (b) of D1.

```
Decode two consecutive instructions: I1 and I2
    If the following are all true
        I1 is a 'simple' instruction
        I2 is a 'simple' instruction
        I1 is not a jump instruction
        Destination of I1 ≠ Source of I2
        Destination of I1 ≠ Destination of I2
    Then issue I1 to U-pipe and I2 to V-pipe
    Else issue I1 to U-pipe
```

(b) Instruction issue algorithm

1. Resource Dependency

- When 2-instructions require a single functional unit or data path, a resource dependency occurs.
- The ????P during the D1 stage issues 2-instructions for parallel execution if both belong to the class of simple instructions, thereby eliminating most resource dependencies.
- The instructions must be directly executed that does not require micro-coded sequencing.

- The instruction being issued to the V-pipe can be an ALU operation, memory referencing or a jump.
- The instruction being issued to the U-pipe can be from the same categories or from an additional set that uses a functional unit available only in the U-pipe, such as the barrel shifter.
- Although the set of instructions identified as 'simple' might seem restrictive, more than 90% of the instructions executed in the integer SPEC benchmark suite are simple.

2. Data dependencies
   - When one instruction writes a result that is read or written by another instruction, a data dependency occurs.
   - Logic in D1 ensures that the source and the destination registers of the instruction issued to the V-pipe differ from the destination register of instruction issued to the U-pipe.
   - This arrangement eliminates read-after-write (RAW) and write-after-write (WAW) dependencies.
   - Write-after-read (WAR) dependencies need not be checked because reads occur in an earlier stage of the pipelines than writes.
   - The design includes logic that enables instruction with certain special types of data dependency to be executed in parallel.
   - For example, a conditional branch instruction that tests the flag results can be executed in parallel with a compare instruction that sets the flags.
3. Control dependencies
   - When the result of one instruction determines whether another instruction will be executed, a control dependency occurs.
   - The ????P in D1 never issues an instruction to the V-pipe when a jump instruction is issued to the U-pipe, thereby eliminating control dependencies.

Q5. more questions are available in Brainheaters app....

.

.

.

Brainheaters

Download the App!