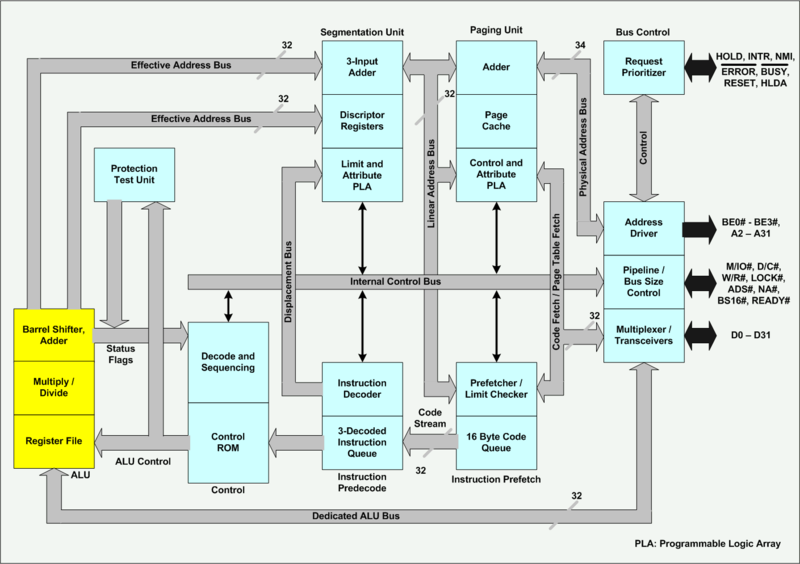
1 –

a)

b) b. Explique a função de cada uma destas unidades.

• Bus Interface Unit (Unidade de barramento): É responsável por controlar os acessos externos à memória e interfaces de e/s. Esta unidade recebe pedidos de acesso à memória de várias outras unidades.

• Prefech Unit (Unidade de pré-busca): Busca antecipada de instruções. Esta unidade possui uma fila de instruções (code queue) de 16 bytes, e solicita acessos à memória para a unidade de barramento sempre que esta fila se encontra parcialmente vazia ou quando uma instrução de transferência de controle é executada.

• Decode Unit (Unidade de decodificação): É responsável por decodificar as instruções. Esta unidade retira uma instrução da fila de instruções e gera uma instrução decodificada que é armazenada na fila de instruções decodificadas (decoded queue).

• Control Unit (Unidade de controle): Retira instruções decodificadas da fila para serem executadas. Esta unidade controla todas as demais unidades que participam na execução da instrução.

• Data Unit (Unidade de dados): É formada por uma ALU e registradores de 32 bits.

• Protection Unit (Unidade de proteção): Verifica se um endereço se encontra dentro dos limites de um segmento.

• Segmentation Unit (Unidade de segmentação) e Paging Unit (Unidade de paginação): Realizam a conversão entre endereço virtual e endereço real. c) O 80386 original possuía tanto interna quanto externamente um barramento de dados e de enderecos de 32 bits.

d) Capacidade máxima de memória de 5 Gbyte. (Tem que explicar isso wtf)

e) Existem os General Registers, Segment Registers e Flags Registers.

General Registers:

* EAX, acumulador usado em operacoes aritméticas.
* EBX, base, usado para apontar para dados no segmento DS.
* ECX, contador, usado em loops.
* EDX, dados, é uma extensão do EAX.
* ESI, índice da fonte de dados a copiar.
* EDI, índice do destino de dados a copiar.
* EBP, apontador da base do frame.
* ESP, apontador da pilha, aponta para o topo da pilha.

Segment Registers:

* CS, code segment (segmento de código).
* SS, stack segment (segmento de pilha).
* DS, data segment (segmento de dados).
* ES, data segment (segmento de dados).
* FS, data segment (segmento de dados).
* GS, data segment (segmento de dados).

Flag Registers:

Flags de status:

* CF, carry flag, indica a existência de um bit de transporte.
* PF, parity, 1 para pariedade e 0 para pariedade impar.
* AF, auxialiary carry, indica a existência do bit transporte ou empréstimo após uma operacão aritmética
* ZF, zero, indica se o resultado de uma operacão aritmética é igual a zero ou diferente de zero.
* SF, sign, indica se resultado de uma operacão com sinal positivo (0), em caso contrario (1).
* OF, overflow, indica em estouro da capacidade de armazenamento de um registrador.

Flags de controle:

* TF, trap, permite a operacão “passo a passo”para debug.
* IF, interrupt enable, indica se as interrupcões estao habilitadas (1) ou (0).
* DF, direction, para operacões com strings, se D=0 os registradores de índice serão incrementados, caso contrário (D=1) serão decrementados.

Flags especificos do 80386:

* IOPL, nivel de privilégio de E/S.
* NT, tarefa alinhada.
* RF, flag resume.
* VM, modo virtual.

f) Os EFLAGS sao os bits de controle presentes no registrador que exibem o estado atual do processador, també auxiliando as operacoes onde haja necessidade, tais como o Flag de reinicio, verificacao de alinhamento, flag de identificacao, flag direcional, flag direcional, flags de privilegio, flag trap, e flag para habilitar interrupcao e junto com esses há também os códigos condicionais tais como:

Carry Flag (C): Que indica a existência de um bit de transporte ou empréstimo na posicao do bit mais a esquerda após uma operacao aritmética.

Overflow Flag(O): Informa um overflow aritmético após uma adicao ou subtracao de aritmética de complemento de dois.

Parity Flag(P): 1 para paridade par e 0 para paridade impar, no byte menos significativo do resultado de uma operacao aritmética.

Auxiliary Carry Flag(A): Usado na aritmética BCD, para indicar a existência do bit de transporte ou empréstimo após uma operacao aritmética entre dois bytes, ou lógica de 8 bits.

Sign Flag(S): Exibe o sinal de uma operacao lógica ou aritmética.

Zero Flag(Z): Literalmente 0 para um resultado de uma operacao aritmética ou lógica.

g)

• Operações aritméticas sobre números inteiros, tais como adição e subtração (ADD, SUB).

• Operações lógicas bit-a-bit, tais como AND e OR.

• Operações de movimentação de dados, que transferem dados entre os registradores ou entre os registradores e a memória principal (MOV).

h) O sistema de enderecamento é representado da seguinte forma segment:offset (segmento e deslocamento) e os dois pontos “:” chamado separador. Os enderecos lógicos diferentes que nos dão um mesmo endereco absoluto na memória são chamados de segmento:offset pares.

i) Depois que ele compila ele gera um endereco lógico, formado por segmento:offset, logo depois ele vai para a unidade de segmentacao passando para o endereco linear formado com 32 bits e finalizando o processo indo para a unidade paginacão formando o endereco fisíco de 32 bits.

j)

• Modo registrador: A instrução indica o número de um registrador de dados onde se encontra um operando (fonte ou destino).

• Modo imediato: S instrução referencia um operando que se encontra dentro do próprio código da instrução.

• Modo implícito: A localização do operando não está explicitamente indicada na instrução. Por exemplo, nas chamadas arquiteturas acumulador, um dos operandos-fonte e o operando-destino nas instruções aritméticas/lógicas encontra-se sempre em um registrador especial, o acumulador. Assim, não é necessário que este registrador seja explicitamente referenciado pela instrução.