Nome: Artur Vítor

3-

a) PUSHI – Coloca um valor imediato na memoria

ADD – Soma os dois valores superiores da memoria e armazena no endereço superior

MUL – Multiplica os dois valores superiores da memoria e armazena no endereço superior

ADDI – Soma o valor superior da memoria com um imediato e armazena no endereço superior

DUP – Duplica o topo da pilha

POP – Remove o valor superior da memoria

INC – Incrementa o topo da pilha

DEC – Decrementa o topo da pilha

JZ - Salta se topo da pilha for 0

J - Salto incondicional

b)

LOAD - Carrega um endereço da memória em ACC

LOADI - Carrega um valor imediato em ACC

STORE – Empilha o valor em ACC na memória (Push)

ADD – Soma o valor em ACC com um valor da memória e armazena em ACC

MUL - Multiplica o valor em ACC com um valor da memória e armazena em ACC

DEC – ACC = ACC - 1

JZ – Se ACC == 0, faz o jump para o endereço

J - Salto incondicional

c)

LOAD – Carrega um valor da memória para Rn

STORE – Guarda na memória o valor de Rn

MOV – Move o valor imediato ou do registrador para Rn

ADD - Soma R1 com R2 e armazena em R1

MUL – Multiplica R1 por R2 e armazena em R1

DEC – Decrementa o valor do registrador especificado

JZ - Se Rn == 0, faz o jump para o endereço

J - Salto incondicional

d)

LOAD R0 - Apenas R0 pode carregar da memória

STORE R0 - Apenas R0 pode armazenar na memória

MOV – Move o valor imediato ou do registrador para Rn

ADD - Soma R1 com R2 e armazena em R1

MUL – Multiplica R1 por R2 e armazena em R1

DEC – Decrementa o valor do registrador especificado

JZ - Se Rn == 0, faz o jump para o endereço

J - Salto incondicional

4 – As instruções podem sem as mesmas do exercício anterior, a diferença será apenas no tamanho dos registradores, memoria e instruções, mas a estrutura das instruções se manterá a mesma.

5 – Opcode, operandos e modo de endereçamento (direto, indireto ou imediato), bits de controle e a próxima instrução

6 –

- Registradores: Dados armazenados na CPU (ex: EAX, R0).
- Memória principal: Endereços de memória (ex: MOV [0x100], AX).
- Pilha (stack): Operandos em estruturas LIFO (ex: PUSH/POP).
- Imediatos: Valores constantes embutidos na instrução (ex: ADD R1, #5) [Apenas origem].
- Dispositivos de E/S: Portas de entrada/saída (em arquiteturas com E/S mapeada).

7-

- Endereço 1: Operando de origem 1 (ex: valor de A).
- Endereço 2: Operando de origem 2 (ex: valor de B).
- Endereço 3: Destino do resultado (ex: armazenar A + B).
- Endereço 4: Endereço da próxima instrução (em máquinas com fluxo explícito).

8-

- Completude: O conjunto deve permitir implementar qualquer operação necessária.
- Tamanho do código: Instruções devem ser compactas para economia de memória.
- Velocidade: Operações frequentes devem ser rápidas (ex: acesso a registradores).
- Regularidade: Padrões consistentes facilitam a decodificação (ex: tamanho fixo).
- Compatibilidade: Suporte a software legado (ex: x86 mantém modos antigos).

9 – Registradores, Imediatos, Memória, Implícitos (Em processadores com operações que ocorrem em um acumulador principal por exemplo)

10 – No deslocamento lógico, o bit descolado é substituído por um zero, diferentemente do deslocamento aritmético onde na operação há a precaução de manter o bit de sinal inalterado.

11 – Instruções de controle de fluxo são necessárias para alterar o fluxo de execução de um programa, sem elas não seria possível criar possível criar loops, chamar funções ou desviar para um outro trecho do programa com base em uma condicional, assim diminuindo substancialmente a flexibilidade do programa.

12 – Flags de status: Teste de bits (ex: ZERO, CARRY) após uma operação (ex: CMP R1, R2).

Valor direto: Comparação explícita (ex: BEQ R1, R2, label).

13 –

0 endereço:

PUSH B ; PUSH C ; MUL ; PUSH A ; ADD ; PUSH D ; PUSH E ; PUSH F ; MUL ; SUB ; DIV ; POP X

1 endereço:

LOAD B ; MUL C ; ADD A ; STORE T1 ; LOAD E; MUL F ; STORE T2 ; LOAD D ; SUB T2 ; STORE T3 ; LOAD T1 ; DIV T3 ; STORE X

2 endereços:

MOVE R1, B ; MUL R1, C ; ADD R1, A ; MOVE R2, E ; MUL R2, F ; MOVE R3, D ; SUB R3, R2 ; DIV R1, R2 ; MOVE X, R1

3 endereços:

MOVE R1, B ; MUL R1, R1, C ; ADD R1, R1, A ; MOVE R2, E ; MUL R2, F ; MOVE R3, R3, D ; SUB R3, R3, R2 ; DIV R1, R1, R2 ; MOVE X, R1

14 – É a capacidade de chamar um procedimento dentro de outro (ex: função A() chama B(), que chama C()). Requer salvamento hierárquico do endereço de retorno (ex: usando pilha).

15 –

- Pilha (stack): Mais comum (ex: x86, ARM).
- Registrador dedicado para armazenar endereço de retorno.
- Memória fixa (Geralmente para sistemas sem suporte a pilha)

16 - É um procedimento que pode ser interrompido e chamado novamente antes de terminar (ex: recursão ou multithreading). Não depende de dados estáticos (ex: usa apenas pilha ou registradores).

17 –

a) MIPS

- **Registradores de Propósito Geral (GPR):** 32 registradores de 32 bits, nomeados de $0 a $31 (ex: $zero, $t0–$t9, $s0–$s7)
- **Registradores Especiais:**
    - PC (*Program Counter*): Armazena o endereço da próxima instrução
    - HI e LO: Usados para armazenar resultados de multiplicações/divisões (64 bits)
    - STATUS: Contém flags de estado (ex: zero, overflow)
    - Sp, fp, cause, epc

b) x86-32 (Intel/AMD 32-bit)

- **GPR:**
    - EAX, EBX, ECX, EDX: Usados para cálculos e transferência de dados
    - ESI, EDI, EBP, ESP : Ponteiros (stack, base, índice)
- **Especiais:**
    - EIP (*Instruction Pointer*): Equivalente ao PC
    - EFLAGS: Flags de status (zero, carry, etc.)
    - ESP (*Stack Pointer*): Topo da pilha

c) x86-64 (64-bit)

- **GPR:** Extensão dos registradores 32-bit para 64-bit (RAX, RBX, ..., R15).
- **Especiais:**
    - RIP (*Instruction Pointer* 64-bit)
    - RFLAGS: Flags estendidas
    - RSP (Stack Pointer 64-bit)
    - Rbp (Base pointer)
    - Registradores de segmento

d) ARMv7

- **GPR:** 12 registradores (R0–R12 para dados)
- **Especiais:**
    - R13 (SP): Stack Pointer
    - R14 (LR): *Link Register* (endereço de retorno de chamadas)

- R15 (PC): Program Counter
- CPSR: Registro de status (flags)
- SPSR (Saved Program Status Register)

e) MOS6502 (Processador clássico, ex: NES)

- **GPR:**
  - A (Acumulador): Principal registrador para operações.
  - X, Y: Registradores de índice.
- **Especiais:**
  - PC (16-bit): Program Counter.
  - SP (8-bit): Stack Pointer
  - P (*Processor Status*): Flags (zero, carry, etc.).

18-

a) x86

- **Imediato:** MOV EAX, 42
- **Registro:** MOV EAX, EBX
- **Direto:** MOV EAX, [0x1000]
- **Indireto:** MOV EAX, [EBX]
- **Indexado:** MOV EAX, [EBX+ESI*4]

b) ARMv7

- **Imediato:** MOV R0, #10
- **Registro:** ADD R0, R1, R2
- **Deslocamento:** LDR R0, [R1, #4]
- **Pré-indexado:** LDR R0, [R1, #4]!
- **Pós-indexado:** LDR R0, [R1], #4

c) RISC-V

- **Imediato:** ADDI x1, x2, 5
- **Registro:** ADD x1, x2, x3
- **Base + offset:** LW x1, 8(x2)

d) Z80

- **Imediato:** LD A, 0x12
- **Indireto:** LD A, (HL)
- **Indexado:** LD A, (IX+5)