#Responsáveis pelo desenvolvimento do projeto

- Projetista -> Concepção -> Dá as instruções e o objetivo ao projeto
- Arquiteto -> Segue as instruções, cria o que foi planejado na medida do possível
- Projetista fórmula e gerência, arquiteto cria o que foi pedido na medida do possível alterando o que for preciso com comunicação ao projetista.

#Software & Hardware

- Tudo que o software pode fazer o hardware também pode.

#Lógica

- lógica generalizada é a forma racional de resolver determinado sendo de forma eficiente ou não, necessitando ação para efetivar a lógica
- Lógica matemática -> Verdadeiro / Falso

#Regra de negócio

- Objetivo, requisitos, CB(custo\tempo\necessidade)
- Baseada nos pilares da Disponibilidade, integridade e confiabilidade
    - Mais importante é a integridade, o segundo é a confiabilidade, caso de ruim, o negócio acaba
    - Quanto mais elevado estes parâmetros, melhor o CB para empresa
    - Quanto maior a confiabilidade maior a fragilidade da empresa
- Quanto maior a confiança no negócio maior a fragilidade

#Programação

- Sequencial -> De um para um
- Corrente -> De um para vários
- Paralela -> Simultâneo -> De vários para vários

#Paralelismo

- Executar mais de uma instrução ao mesmo tempo, Ex: Pipeline
- Pode ser obtido em vários níveis com ou sem uso de programação
- Níveis
    - Instrução -> Granulosidade fina -> Utilizado em pipeline, superescalar, VLIM
    - Tarefas -> Granulosidade média -> Arquitetura SMT
    - Processos -> Granularidade grossa -> Computação paralela utilizando multiprocessadores e multicomputadores

#Classificação do paralelismo

- Flynn(1972) -> Mais conhecido e baseia-se na unidade de multiplicidade do fluxo de dados e instruções
- Duncan(1990) -> Menos conhecida e classificação mais recente e abrangente

#Arquiteturas baseadas em Flynn

- SISD: Single Instruction stream, Single Data stream -> Fluxo de instrução única, fluxo de dados único

    - Instruções são executadas serialmente, porém em estágios com uso do pipeline e estes podem ser sobrepostos;
    - Sabe exatamente o fluxo da instrução;
    - 1 processador controla 1 U.C e executa 1 instrução
- MISD: Multiple Instruction stream, Single Data stream -> Fluxo de várias instruções, fluxo de dados único
    - Conjunto de processadores para processar único dado.
    - N processadores que dividem uma mesma memória
- SIMD: Single Instruction stream, Multiple Data stream -> Fluxo de instrução única, fluxo de dados múltiplos
    - Representa processamento matricial, paralelo e associativo
    - Única informação é transmitida para múltiplos processadores que trabalham de forma independente um ao outro.
    - N processadores controlados por 1 U.C sobre N fluxos
- MIMD: Multiple Instruction stream, Multiple Data stream -> Fluxo de várias instruções, vários fluxos de dados
    - Vários processadores, onde cada um é controlado por uma U.C, onde recebem instruções diferentes e operam sob fluxo de dados diferentes
    - Processos podem ser síncronos ou assíncronos
    - N processadores controlados por N U.C's sobre N fluxos.
- MIMD engloba tudo, derivado de SIMD e MISD gera o SISD

#Modelos de acesso a memória
- Cada lugar da memória possui um endereço que inicia em 0 e vai até 2n - 1, onde n é o número de bits do endereço
- Modelo convencional consiste em processador, executando programa em memória
- Memória compartilhada -> Multiprocessadores
- Memória distribuída -> Multicomputadores

#MIMD com memória compartilhada
- Memória é acessada por múltiplos processadores
- Sincronização entre tarefas é feita por escrita/leitura na/da memória compartilhada e usuário é responsável por sua especificação
- O uso desses modelos de arquitetura possuem R.C em memória
- Arquitetura BlackBoard
- Comunicação entre tarefas é rápida
- Escalabilidade limitada pelo hardware

#MIMD com memória distribuída - > Server para V.M também
- Memória física distribuída entre processadores reservadas ao mesmo
- Tarefas se comunicam através de troca de mensagens e sincronização
- Bibliotecas possuem rotinas de comunicação e programas sequências são bastante utilizadas
    - Problema dividido em número de tarefas que se comunicam
    - **MPI**(Message Passing Interface) é um protocolo independente de linguagem usado para programar computadores paralelos.

    - **MPP** (Massively Parallel Processors) é interconectado os processos por rede de alta velocidade, possui boa escalabilidade e complicado para programar
    - **COW** (Cluster of Workstations) utilizado em terminal com processamento local em rede e tem boa relação custo/benefício

#Endereçamento
- Direto -> Posição de memória -> Dinâmico
- Imediato -> Valor estático imputado ou fixo -> Estático

#Instruções em quantidades de endereços
- Quanto mais endereços mais fácil a lógica, mais fácil manusear, porém é menos otimizado
- Quanto menos endereços, mais difícil manusear e é mais otimizado

#Instrução

| E1 | ADD BCA E2 | Faz a soma de B + C que resulta em A e grava no endereço E2 de memória

- Operação E1
- ADD -> Operação a ser realizada
- BCA -> Variável B + C resulta em A
- A será gravado na posição de memória E2
- Exemplo acima é uma instrução de 4 endereços onde B - 1, C - 2, A - 3 e E2 - 4 -> 4, 4 bits de instrução de memória
- Quanto menor a quantidade de memória mais otimizada é a instrução e fica mais complicado de se fazer a programação
- Caso diminuirmos a quantidade de endereços podemos diminuir a quantidade de instruções e otimizar em uma única tarefa com vários processos internos.

#Baixo nível
- Assembly
    - Não proprietária, nome é associação a montagem de hardware
    - Um passo acima do mais baixo(binário)
    - Pode manipular componentes e fluxos, depende do nível de acesso ao S.O
- Assembler
    - Semelhante a um compilador, faz a conversão de caracteres para baixo nível
    - Ex: MASM -> Microsoft, GAS -> Unix, NASM -> Multi.S.O
- Os comandos não mudam, o que muda é sistema e arquitetura

#Diferença entre assemblers
- NASM e MASM algumas vezes usam sintaxe Intel, enquanto GAS usa o AT&T
- GAS usa % como prefixo para registros
- GAS, recurso primeiro, destino por último, NASM e MASM ao contrário
- GAS precisa de notações de tamanho nas instruções (b, w, l sufixos) e alguns outros operadores

- GAS usa $ para resultados imediatos, e também variáveis
- GAS usa rep/repe/repne/repz/repnz, prefixos de separação
- MASM escreve registradores FPU como ST(0), ST(1), etc
- NASM é case-sensitive, MASM não

#Manipulação baixo nível
- Controle sobre registradores(Depende se é a nível de programa ou S.O), regiões de memória e a ULA, além do controle total sobre outros componentes

#Registradores
- Constituídos por N flip-flops e cada um consegue armazenar 1 bit
- 2 tipos
    - Geral -> Uso em geral do sistema e é totalmente manipulável pelo usuário
    - Específico -> Uso do funcionamento interno do CPU, mais baixo nível que o geral.

#Evolução de 32 a 64 bits
- Operações 32 rodam em 64, mas não ao contrário sem manipulação da instrução
- Alteração da nomenclatura dos registradores de E para R, aumento na quantidade de registrados além de novos implementados

---

Segundo semestre

#Nível de arquitetura de conjunto de instruções (nível convencional de máquina):
- Visão geral do nível(ocorrência em determinada camada do sistema)
- Tipos de dados(Até que ponto descer para operar um dado)
- Fluxo de controle das instruções do nível
- Programas em alto nível são traduzidos para a linguagem intermediária ISA(Nível de compilador), para então o hardware entender as instruções. Alto nível passa para assembly na ISA e o hardware ai entende o programa.

#Nível de arquitetura de conjunto de instruções:
- Formatos de instruções;(Funções de controle do sistema e circuitos)
- Endereçamento.(Regiões de memória)

#Ciclo de busca/decodificação/execução da U.C
- Busca na memória da próxima instrução a ser executada.
- Decodificação do opcode.
- Leitura dos operandos da memória, se necessário.
- Execução da instrução e armazenamento dos resultados.
- Volta ao primeiro tópico

#Nível de arquitetura de conjunto de instruções:
- Tipos de instruções;
- Fluxo de controle.

- Operações, escalonamento, gerenciamento de hardware, hierarquia...

#Nível de máquina de S.O:
- Memória virtual (Endereçamento virtual para o endereçamento físico, troca, paginação, região ativa, região crítica)
- Fornecedor todas as ferramentas para comunicação do software pelo hardware
- Sistema de arquivos, escalonamento de processos, gerenciamento de memória, entre outros mais recursos.

#Nível de máquina de S.O: Instruções virtuais de E/S;
- Instruções virtuais para processamento paralelo.
- Ativamento de Threads
- Uso de arquiteturas como a flynn
- Níveis de tratamento da E/S
    - Abstração do E/S para o usuário
    - Independência ao dispositivo
    - Drivers
    - Controladora
    - Hardware

#Nível de linguagem de montagem (assembly):
- Linguagem de baixo nível que depende de um sistema operacional para fazer a comunicação, sendo dessa forma, cada S.O tem seu próprio jeito de se comunicar com o hardware através da montagem

#Assembly e Assembler
- Tradutores -> Converte linguagem atual para linguagem fonte
- Montador -> linguagem fonte é linguagem de montagem -> Assembly
- Compilador -> Linguagem fonte é de alto nível, converte para baixo nível -> Assembler
- Fonte se torna mnemônicos quando convertido e após passar pelo compilador se torna opcodes, este é o entendido pelo hardware.

#Arquiteturas avançadas de computadores (nível introdutório):
- Projeto de computadores paralelos: modelos de comunicação, redes de interconexão, desempenho, software. -> Clusterização
- Processadores array e vetoriais. -> Arquiteturas
- Arquiteturas inferior a 1 instrução por ciclo -> CISC, RISC
- Arquiteturas igual ou superior a 1 instrução por ciclo -> SuperEscalar, VLIM, SMT e Multicore

#Arquiteturas avançadas de computadores (nível introdutório):
- Multiprocessadores com memória partilhada -> Gerenciamento de memória multicore
- Multicomputadores com passagem de mensagem. -> Cluster

#MD5 -> Mão única

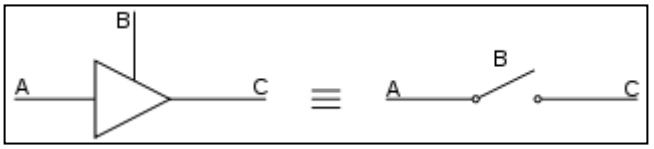
- É somente possível fazer a criptografia, não é possível descriptografar -> Unidirecional
- Utiliza o hash padrão de 128, podendo utilizar de 256, 512
- Passos:
    - Recebe um valor
    - Divide o valor em blocos de 512 bits cada, onde 64 bits são inseridos no final do último bloco, esses 64 bits são usados para armazenar o tamanho do input original.
    - Passa por um buffer com quatro palavras de 32 bits cada, chamadas de A, B, C e D sendo inicializadas nesta ordem
    - Passa por uma tabela K com 64 elementos. sendo numérico i é
    - indicado como Ki. A tabela é montada antes antecipadamente para acelerar os cálculos, sendo utilizada a equação -> Ki = abs(sin(i + 1)) * 232
    - Após isso passa por quatro funções auxiliares em que cada uma recebe três palavras de 32 bits como parâmetro, produzindo uma palavra de 32 bits, elas aplicam operadores lógicos AND, OR, NOT e XOR aos bits de input.
    - Por fim é passado pelo processamento em blocos, os buffers passam pelas auxiliares, assim realizando 16 operações totais por interação.

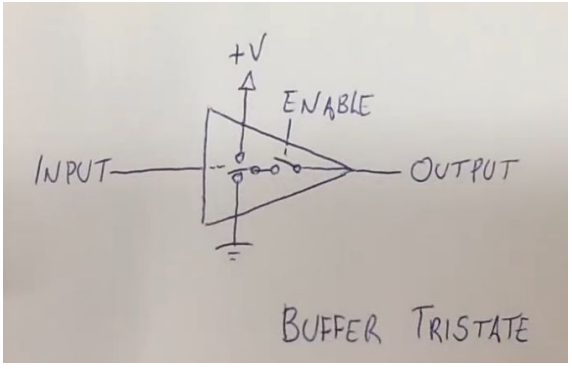
#RSA -> Mão dupla

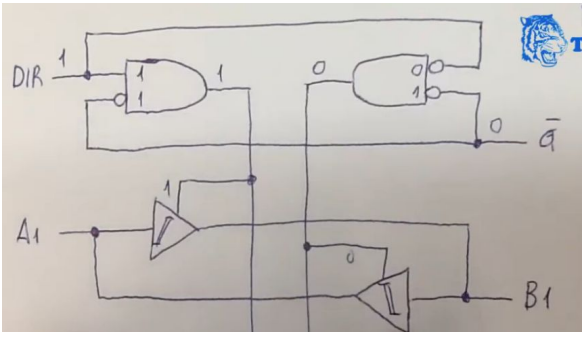
- Chave assimétrica
- Criptografia baseada na segurança da matemática clássica
- Segurança garantida por motivos da fatoração do valores a decriptografar
- Primeira criptografia que tornou possível o uso do certificado digital
- Podemos utilizar uma tabela de conversão dos valores caracteres para valores numéricos ou customizar esta seleção
- É matematicamente possível fazer a criptografia e descriptografia -> Bidirecional
- Passos:
    - Utiliza valores primos diferentes um do outro P e Q a ^100 unidades de tamanho
    - Realiza a equação de N -> N = P * Q
    - Realiza a equação toteliente ou Phi -> Q(N) = (P - 1) * (Q - 1)
    - Realiza o cálculo de MDC para descobrir o valor co-primo entre MDC(E, Q(N)) = 1 para descobrir o valor de E -> 1 < E < Q(N)
    - Faz o inversamente multiplicável de E para resultar no valor D
    - Após isso temos as chaves:
        - Pública (N, E)
        - Privada (N, D)
    - Para criptografar usamos: RSA = C^E mod (N)
    - Para decriptografar usamos: C = RSA^D mod (N)
- Após os passos acima, podemos realizar a criptografia e descriptografia

#Tri-state

- Adiciona um estado isolado a estrutura
- Multiplexador de 3 estados, onde dois estados são os comuns e um terceiro é uma resistência para os barramentos e pode ser implementado como um buffer para outros que fazem conexão dentro do barramento.
- Tem que serem controlados para não causar danos tanto lógicos quanto físicos ao sistema.







Os triângulos são tri-states, isso é o B1 que está enviando o sinal 0, está isolado do A1. Neste exemplo o sinal está trafegando de A para B, se inverter a entrada do DIR para 0, o sinal irá correr de B para A.

#Estados possíveis do buffer acima:

- DIR -> 0 com A -> 0, o buffer ficará carregado com o lado de B para A
- DIR -> 1 com A -> 0, o buffer ficará carregado com o lado de A para B
- DIR -> 0 com A -> 1, o buffer ficará vazio, os dois lados ficaram limpos
- DIR -> 1 com A -> 1, o buffer ficará vazio, os dois lados ficaram limpos

#PLC

- Criptografia a nível de hardware, operações a mais baixo nível que que se pode oferecer antes do binário.

#Endereçamento

UNIP UNIVERSIDADE PAULISTA
Rua Miguel Guidote, 405 - Egisto Ragazzo, Limeira - SP, 13480-290
Aluno: Jonathan Jonys Miranda
Curso: Ciências da Computação
RA: C4400D-2

06/09/2017

Arquiteturas

⇩

Baseado em instrução

⇩

Aritmética

⇩

0,1,2,3 e 4

Endereços...

Expressão:

$A \leftarrow ((B + C) * D + E - F / (G * H))$

↑

Aritmética

Solução utilizando 4 endereços

ENDEREÇO	INSTRUÇÃO			COMENTÁRIO
E1	ADD	BCA	e2	Soma B com C, resultado em A, vai para e2
E2	MUL	ADA	e3	Multiplica A por D, resultado em A, vai para e3
E3	ADD	AEA	e4	Soma A com E, resultado em A, vai para e4
E4	SUB	AFA	e5	Subtrai F com A, resultado em A, vai para e5
E5	DIV	AGA	e6	Divide A por G, resultado em A, vai para e6
E6	DIV	AHA	e7	Divide A por H, resultado em A, vai para e7
E7	HALT			Fim do Programa

Solução utilizando 3 endereços

ENDEREÇO	INSTRUÇÃO		COMENTÁRIO
E1	ADD	BCA	Soma B com C, resultado em A, incrementa PC
E1+1	MUL	ADA	Multiplica A por D, resultado em A, incrementa PC
E1+2	ADD	AEA	Soma A com E, resultado em A, incrementa PC
E1+3	SUB	AFA	Subtrai F com A, resultado em A, incrementa PC
E1+4	DIV	AGA	Divide A por G, resultado em A, incrementa PC
E1+5	DIV	AHA	Divide A por H, resultado em A, incrementa PC
E1+6	HALT		Fim do Programa

15/09/2017

Expressão:

$$A \leftarrow ((B + C) * D + E - F / (G * H))$$

Aritmética

Solução utilizando 2 endereços

ENDEREÇO	INSTRUÇÃO		COMENTÁRIO
E1	MOV	AB	Move B para A
E1+1	ADD	AC	Soma A com C, resultado em A
E1+2	MUL	AD	Multiplica A por D, resultado em A
E1+3	ADD	AE	Soma A com E, resultado em A
E1+4	SUB	AF	Subtrai F de A, resultado em A
E1+5	DIV	AG	Divide A por G, resultado em A
E1+6	DIV	AH	Divide A por H, resultado em A
E1+7	HALT		Fim do Programa

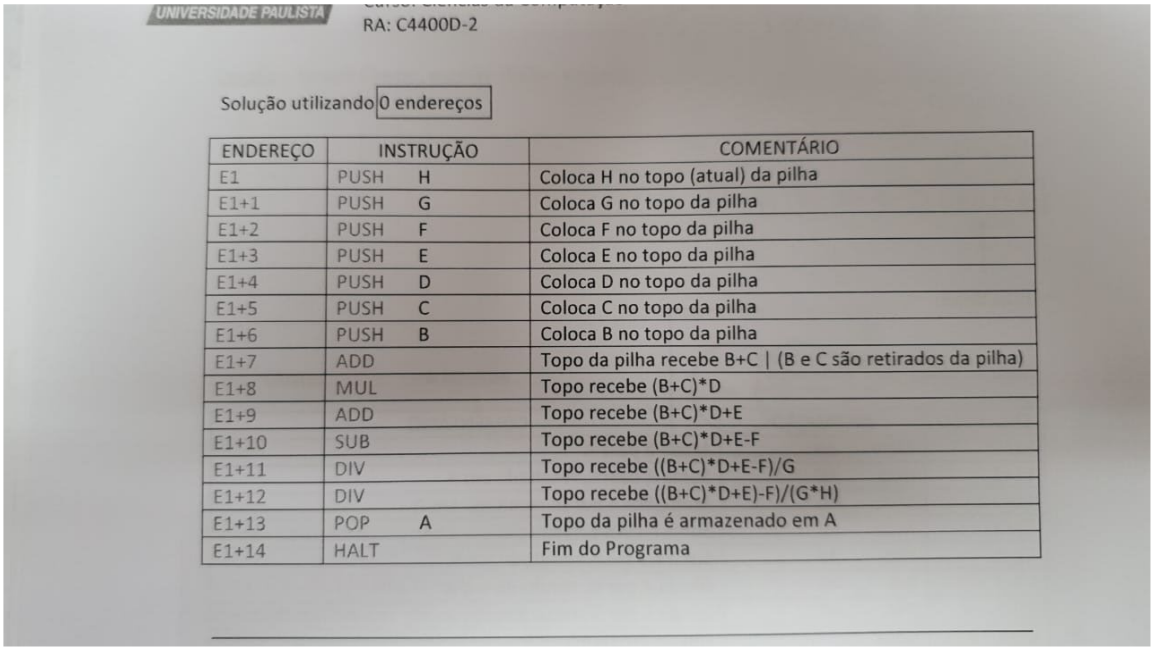
Solução utilizando 1 endereço

ENDEREÇO	INSTRUÇÃO		COMENTÁRIO
E1	LDA	B	Move B para o acumulador
E1+1	ADD	C	Soma acumulador com C, resultado no acumulador
E1+2	MUL	D	Multiplica acumulador por D, resultado no acumulador
E1+3	ADD	E	Soma acumulador com E, resultado no acumulador
E1+4	SUB	F	Subtrai F do acumulador, resultado no acumulador
E1+5	DIV	G	Divide acumulador por G, resultado no acumulador
E1+6	DIV	H	Divide acumulador por H, resultado no acumulador
E1+7	STA	A	Armazena acumulador no endereço de A
E1+8	HALT		Fim do Programa

OBS.:

- Requer acumulador
- Requer instruções
  - Load
  - Store

Para mover ou copiar dados na memória.



UNIVERSIDADE PAULISTA

RA: C4400D-2

Solução utilizando 0 endereços

ENDEREÇO	INSTRUÇÃO		COMENTÁRIO
E1	PUSH	H	Coloca H no topo (atual) da pilha
E1+1	PUSH	G	Coloca G no topo da pilha
E1+2	PUSH	F	Coloca F no topo da pilha
E1+3	PUSH	E	Coloca E no topo da pilha
E1+4	PUSH	D	Coloca D no topo da pilha
E1+5	PUSH	C	Coloca C no topo da pilha
E1+6	PUSH	B	Coloca B no topo da pilha
E1+7	ADD		Topo da pilha recebe B+C | (B e C são retirados da pilha)
E1+8	MUL		Topo recebe (B+C)*D
E1+9	ADD		Topo recebe (B+C)*D+E
E1+10	SUB		Topo recebe (B+C)*D+E-F
E1+11	DIV		Topo recebe ((B+C)*D+E-F)/G
E1+12	DIV		Topo recebe ((B+C)*D+E)-F)/(G*H)
E1+13	POP	A	Topo da pilha é armazenado em A
E1+14	HALT		Fim do Programa

#Links

http://www.inf.furb.br/~maw/arquitetura/aula11.pdf

https://www.gta.ufrj.br/ensino/EEL580/apresentacoes/Parte2.pdf

http://www.inf.furb.br/~maw/arquitetura/aula14.pdf

https://dcc.ufrj.br/~gabriel/arqcomp2/Avancadas.pdf

https://www.dca.ufrn.br/~pablo/FTP/arq_de_comp/apostilha/capitulo6.pdf