Pedro Spoljaric Gomes

Projeto e Desenvolvimento de um Sistema Computacional - Sistema Operacional

São José dos Campos - Brasil

janeiro de 2021

Pedro Spoljaric Gomes

Projeto e Desenvolvimento de um Sistema Computacional - Sistema Operacional

Relatório apresentado à Universidade Federal de São Paulo como parte dos requisitos para aprovação na disciplina de Laboratório de Sistemas Computacionais: Sistemas Operacionais.

Docente: Prof. Dr. Tiago de Oliveira

Universidade Federal de São Paulo - UNIFESP

Instituto de Ciência e Tecnologia - Campus São José dos Campos

São José dos Campos - Brasil

janeiro de 2021

Resumo

Este trabalho descreve o projeto de um sistema operacional capaz de operar no processador desenvolvido com base na arquitetura criada, com o principal objetivo de executar e gerenciar algoritmos criados e compilados usando o compilador para linguagem C- desenvolvido. O trabalho descreve todo o planejamento feito para o desenvolvimento do sistema operacional, com ênfase nos gerenciamentos de processos, memória, arquivos e entrada/saída, sendo esta etapa muito importante para que o desenvolvimento ocorra de forma mais rápida e consistente.

Palavras-chaves: FPGA, verilog, sistema computacional, processador, sistema operacional, C-.

Lista de ilustrações

Sumário

1 Introdução

A importância da arquitetura de um computador se consolida já no processamento mais básico realizado nele, onde valores de tensão em transistores são convertidos em *bits* e interpretados pelo processador como dados de entrada que devem ser executados. A arquitetura é o modelo que define como esses dados serão interpretados, o que será feito com eles e para onde vão, como um simples comando que realiza a soma entre dois números e armazena o resultado. De acordo com Alan Clements em *Principles of Computer Hardware*, "A arquitetura descreve a organização interna de um computador de um jeito abstrato; isto é, ela define as capacidades de um computador e seu modelo de programação. Você pode ter dois computadores que foram construídos de maneiras diferentes com tecnologias diferentes, mas com a mesma arquitetura" (1, p. 1, tradução nossa).

Outro componente muito importante no desenvolvimento de um sistema computacional é um compilador de linguagem de alto nível, que fará o processo de tradução dos comandos para a linguagem de baixo nível criada. Com ele, é possível trabalhar em diversas partes do sistema num nível de abstração mais alto, deixando o processo mais rápido e agradável.

Finalmente, o que será apresentado neste relatório é o projeto de um sistema operacional baseado na arquitetura e usando o compilador desenvolvidos anteriormente. Este componente de extrema importância é responsável por todo o gerenciamento de execução de processos no processador, o que torna possível o desenvolvimento de programas mais complexos e que podem ser executados em concorrência.

2 Objetivos

2.1 Geral

Projetar um sistema operacional baseado no processador e usando o compilador desenvolvidos anteriormente, que seja capaz de gerenciar diferentes processos simultaneamente.

2.2 Específicos

- Definir como será feito o gerenciamento de processos;
- Definir como será feito o gerenciamento de memória;
- Definir como será feito o gerenciamento de sistema de arquivo;
- Definir como será feito o gerenciamento de dispositivos de entrada e saída;
- Definir as técnicas que serão utilizadas;
- Definir os algoritmos que serão utilizados.

3 Fundamentação teórica

Para projetar um sistema operacional, é necessário vasto conhecimento referente a diversos assuntos da área de sistemas computacionais, os quais foram acumulados no decorrer do curso até esse momento. Os conceitos mais importantes serão apresentados a seguir.

3.1 Arquitetura

De maneira geral, arquitetura de um computador é a definição de seus conceitos e fundamentos de mais baixo nível operacional , ou seja, é a base de todo o projeto de um sistema computacional, incluindo a escolha das peças físicas que serão utilizadas, a organização dos componentes, a linguagem que será interpretada pelo computador, o modo como o computador os interpreta, entre muitas outras coisas que consolidam a base estrutural e funcional do sistema. (2)

3.2 Processador

Segundo o Prof. Jean Galdino (3), o processador, ou também conhecido como CPU, é um circuito integrado programável que realiza as funções de cálculo e tomada de decisão de um computador, sendo o componente mais complexo e mais importante de um dispositivo. Sua estrutura básica contém uma unidade de controle, uma unidade de processamento, e um banco de registradores.

3.2.1 Banco de registradores

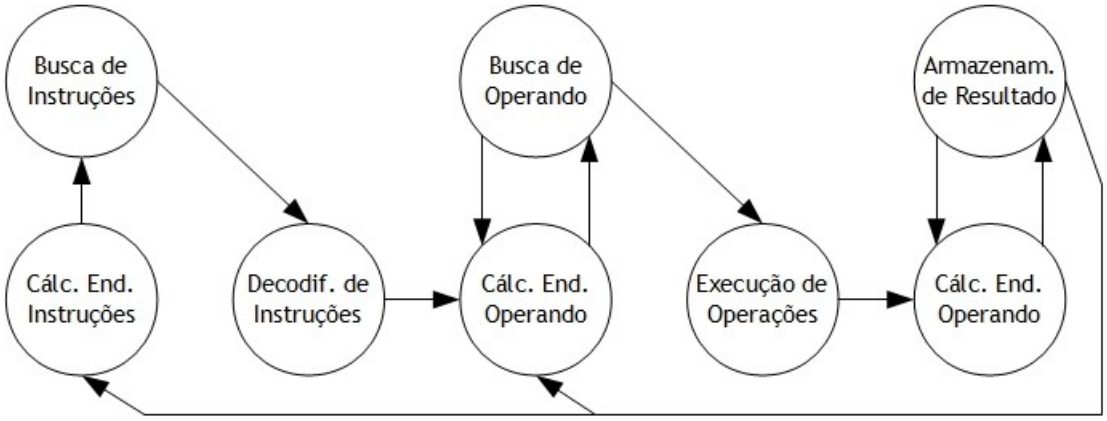
Um processador contém um banco de registradores, que são circuitos capazes de receber informações, guardá-las e transferi-las na direção de algum dispositivo de controle. Segundo o Prof. Raimundo Filho (4), os registradores são algumas posições de memória com as mesmas características das palavras da memória principal.

3.3 Conjunto de instruções

Uma instrução é a interface entre o *hardware* e o *software*, ou seja, é a parte visível para o programador, sendo formada por campos que guardam os dados necessários para a execução da instrução, geralmente apresentados como: Código de Operação e Campos de Endereço (5).

Os Códigos de Operação, ou *opcodes*, indicam a operação que vai ser executada, tanto quanto o modo de endereçamento de operandos utilizado. E os Campos de Endereço indicam o local de cada operando, podendo variar de acordo com a necessidade da operação. Quando uma instrução é apresentada ao processador, ele ativa cada um de seus componentes internos para realizar uma parte da execução desta instrução. Esse processo é demonstrado na Figura 1.

Figura 1 – Diagrama do ciclo de instrução



Fonte: Caderno *Geek* (5)

3.4 Modos de endereçamento

Segundo William Stallings (6), "o campo ou os campos de endereço em um formato de instrução típico são relativamente pequenos. Para que pudéssemos referenciar um grande intervalo de locais da memória principal - ou, em alguns sistemas, da memória virtual -, uma variedade de técnicas de endereçamento foi empregada." Os modos de endereçamento utilizados são: imediato, direto, por registrador e por registrador base.

3.5 *Quartus Prime*

O *software* utilizado para a implementação e simulação neste trabalho será o *Intel Quartus Prime Design* da Intel (7). Ele foi projetado para FPGAs, SoCs e CPLDs Intel, desde a entrada e síntese do projeto até a otimização, verificação e simulação. Sua capacidade foi drasticamente aumentada comparado com as versões anteriores e possui vários elementos lógicos fornecendo aos projetistas a plataforma ideal para aproveitar as oportunidades de design da próxima geração. A versão *Lite* é gratuita e está disponível para *Linux* e *Windows*.

3.6 Linguagem de Descrição de *Hardware*

Uma alternativa à entrada esquemática de um circuito digital em um sistema de projeto auxiliado por computador é utilizar a técnica de projeto de dispositivos lógicos programáveis (PLDs) com uma ferramenta de projeto baseado em texto ou linguagem de descrição de *hardware* (HDL). Exemplos de HDLs são o AHDL - *Altera Hardware Description Language* , e os padrões VHDL e Verilog (8). A linguagem utilizada neste trabalho será o Verilog.

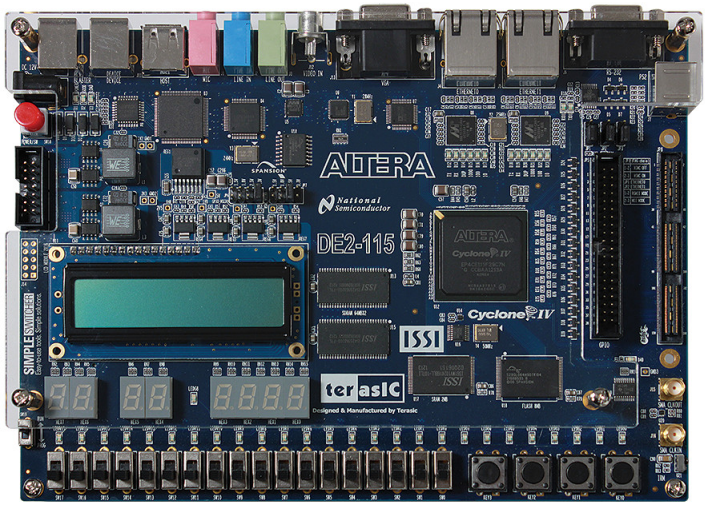
Uma das características de uma linguagem de descrição *hardware* é que ela é independente do *hardware*, ou seja, compatível para ser empregada em diferentes dispositivos.

3.7 FPGA - *Field-Programmable Gate Array*

FPGA, ou *Field-Programmable Gate Array*, é um circuito integrado reprogramável. Segundo Fabbrycio Cardoso da UNICAMP, "um chip FPGA contém um grande número de unidades lógicas que podem ser interconectadas através de uma matriz de trilhas condutoras e de *switches* programáveis". Cada unidade lógica de uma FPGA pode ser configurada de forma independente, o que torna possível modificar o circuito caso o *hardware* sofra alterações.

O kit FPGA utilizado para o desenvolvimento e simulação do projeto foi o *Altera DE2-115 Development and Education Board* mostrado na Figura 2, que contém o FPGA *Altera Cyclone IV E: EP4CE115F29C7* (9).

Figura 2 – FPGA



Fonte: *Altera DE2-115.* (9)

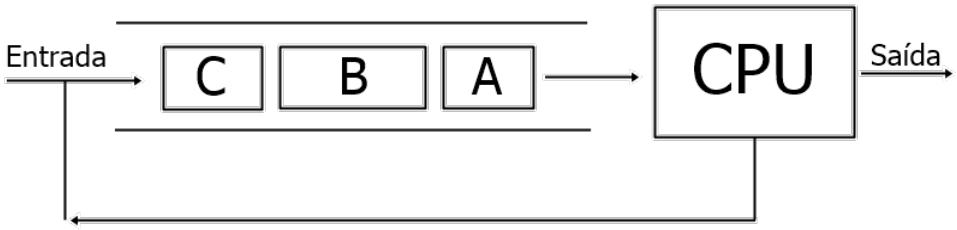
3.8 Preempção

Preempção é como é chamada a interrupção forçada de um processo em execução, ou seja, retirar o controle de um processo sobre A CPU para que outro com maior prioridade possa ser executado. O momento de preempção costuma estar relacionado ao intervalo de tempo que o processo manteve o controle da CPU, denominado de *quantum*.(10)

3.9 Escalonamento

Sempre que ocorre a preempção de um processo, um algoritmo é chamado para que seja decidido qual será o próximo processo a ser executado, este é chamado de algoritmo escalonador. Um exemplo simples de algoritmo escalonador é o escalonamento circular (*Round-Robin*), em que todos os processos prontos para execução formam uma fila e voltam para o final da fila depois de serem executados caso não tenham sido finalizados. A Figura 3 ilustra o funcionamento do algoritmo.(10)

Figura 3 – *Round-Robin*



Fonte: Deinfo. (10)

3.10 Acesso direto à memória (DMA)

Acesso direto à memória, ou DMA, é um método de gerenciamento de entrada e saída em que os dados são escritos e lidos através de um módulo no processador chamado DMAC (controlador de DMA), que realiza a conexão direta de endereços da memória principal à um dispositivo de entrada ou saída externo. Com o uso dessa técnica, é possível realizar a entrada e saída de dados a qualquer momento sem depender da ordem de execução das instruções. (11)

4 Desenvolvimento

O desenvolvimento do sistema operacional será baseado em quatro grupos principais que serão apresentados com detalhes neste capítulo, onde serão apresentados técnicas e algoritmos a serem utilizados no gerenciamento de processos, memória, sistema de arquivos e entrada/saída. Para que esses novos conceitos sejam aplicados, serão necessárias alterações no processador e no compilador, que serão descritas depois dos novos conceitos para que seja estabelecido um contexto.

Além das alterações no processador e no compilador e da introdução dos conceitos de gerenciamento de processos, memória, arquivos e entrada/saída, precisarão ser criados algoritmos para gerar trechos de código intermediários para auxiliar na comunicação entre todas as partes. Um desses algoritmos será feito para atuar entre a preempção e a tomada de controle pelo sistema operacional; o papel desse algoritmo vai ser copiar todos os registradores do banco de registradores para posições específicas da memória onde o sistema operacional tem acesso, assim o sistema operacional pode acessar todos os dados que precisam ser realocados como descrito na seção de gerenciamento de memória.

4.1 Gerenciamento de processos

Ao se planejar como será o gerenciamento de processos pelo sistema operacional, é preciso pensar principalmente em duas coisas: como interromper um processo em execução e como escolher processo que será executado em seguida. A interrupção de um processo em execução é chamada de preempção e o algoritmo que escolhe qual processo será executado em seguida é chamado de escalonador, e esses dois conceitos serão apresentados a seguir.

4.1.1 Preempção

A preempção do processo em execução será feita diretamente pelo processador, que fará a interrupção depois de uma determinada quantidade de ciclos de *clock*, ou seja, cada processo terá acesso ao processador por um tempo limitado antes de ocorrer a preempção. Quando o processador interrompe um processo, ocorre a troca de contexto, isto é, a concessão do controle do processador para outro processo, para que o sistema operacional retome o controle e possa executar o algoritmo escalonador. A troca de contexto também envolve o gerenciamento da memória que foi alocada pelo processo interrompido, e isso será tratado posteriormente numa seção.

4.1.2 Escalonador

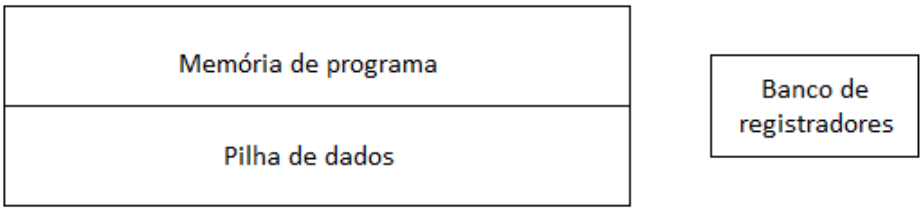
Quando o processador realiza a preempção, o sistema operacional é acionado e este pode executar o algoritmo escalonador para decidir qual processo será executado em seguida. O objetivo desse algoritmo é fazer uma distribuição justa entre os processos para que não ocorra em nenhum deles o que é chamado de inanição, isto é, o processo nunca ser executado. Para fazer o escalonamento, o algoritmo escolhido foi o *Round-Robin*, um algoritmo simples que executa todos os processos ciclicamente, ou seja, sempre que um processo é interrompido por meio de preempção, ele volta para o fim da fila de execução e será executado depois de todos os demais.

4.2 Gerenciamento de memória

Planejar o gerenciamento de memória do sistema operacional envolve diversos desafios, principalmente quando se trata de troca de contexto, pois para que isso aconteça é necessário que toda memória em uso pelo processo que foi interrompido seja armazenada em um lugar específica para que possa ser resgatada quando o processo for retomado. Além disso, é necessário que o sistema operacional ocupe uma parte da memória para armazenar todas as informações de gerenciamento necessárias, como quais processos estão em execução, a fila de processos do algoritmo escalonador e o contador de programa em que cada processo parou de ser executado.

Foi decidido que a memória principal vai ser dividida em alguns blocos. Antes da introdução do sistema operacional, o banco de registradores era isolado e a memória principal era dividida entre memória de programa e pilha de dados, como mostra a Figura 4, de forma que o processo poderia armazenar dados em qualquer trecho da memória e a pilha era controlada pelas instruções desse tipo.

Figura 4 – Arquitetura da memória antes da introdução do sistema operacional

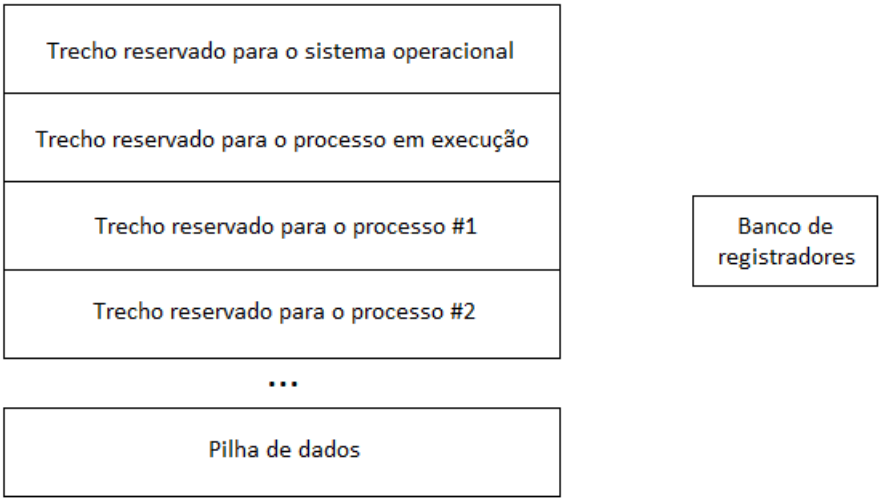


Fonte: O Autor

Com a introdução do sistema operacional, o banco de registradores continua isolado e a pilha continua funcionando da mesma forma, mas o primeiro bloco da memória principal é reservado para o sistema operacional, onde são armazenadas informações como quais programas estão sendo executados e quais são seus contadores de programa atuais, apenas

segundo bloco é acessível pelo processo em execução e os blocos seguintes são trechos reservados para que seja feita a cópia dos dados de programas que sofrerem preempção, como mostra a Figura 5.

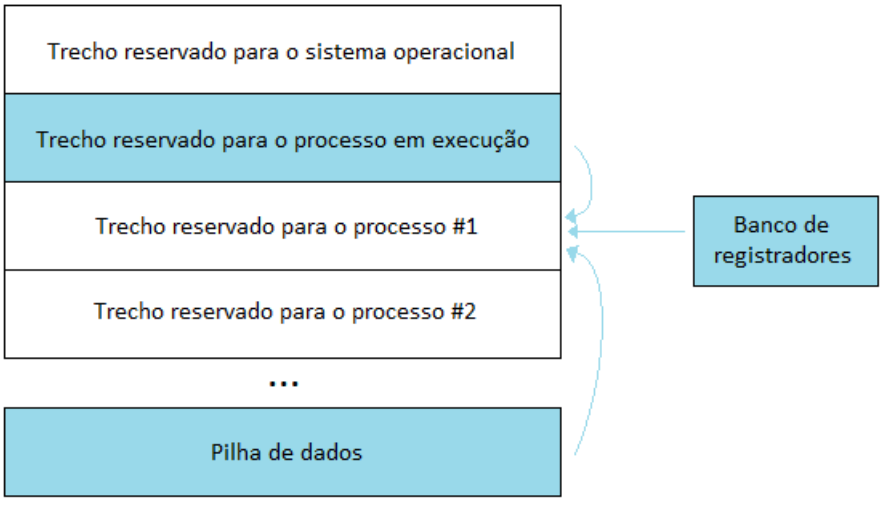
Figura 5 – Arquitetura da memória depois da introdução do sistema operacional



Fonte: O Autor

Quando ocorre troca de contexto, todos os dados do processo atual, destacados em azul na Figura 6, são copiados para o trecho de memória reservado pelo sistema operacional para aquele processo. As setas na figura indicam o que acontece quando o processo #1 que esta em execução é interrompido, os dados da memória, da pilha e do banco de registradores são copiados para o trecho de memória reservado para o processo #1.

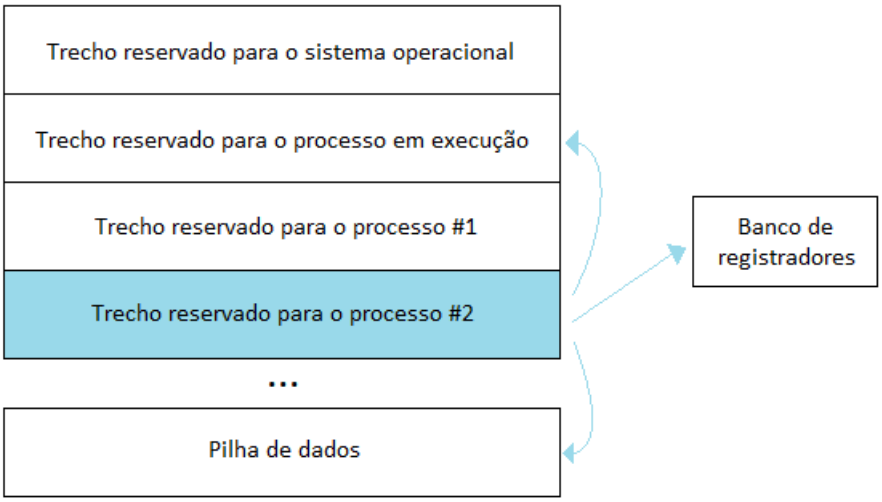
Figura 6 – Realocação dos dados do processo interrompido



Fonte: O Autor

De forma análoga, também ocorre o processo inverso na segunda etapa da troca de contexto, que é quando um processo é selecionado para ser executado. Quando isso acontece, os dados da memória reservada são extraídos e distribuídos entre o trecho de memória do processo atual, a pilha e o banco de registradores, como mostra a Figura 7, mostrando um exemplo de quando o processo #2 é selecionado para execução pelo algoritmo escalonador.

Figura 7 – Realocação dos dados do processo selecionado para execução



Fonte: O Autor

4.3 Gerenciamento de sistema de arquivos

Para criar o sistema de arquivos, será criada uma nova seção de memória que será responsável pelo armazenamento de arquivos binários. Essa seção será composta de vários segmentos de memória de 4 *bytes* (32 *bits*). Para fazer o gerenciamento desses arquivos, o sistema operacional vai possuir três vetores para armazenar nome, endereço inicial e endereço final do arquivo, de modo que um mesmo índice corresponde às informações do arquivo nos três vetores. Serão adicionadas novas instruções para armazenar e carregar um segmento de arquivo em determinada posição da nova memória, de modo similar ao que é feito com o gerenciamento da memória principal.

4.4 Gerenciamento de entrada e saída

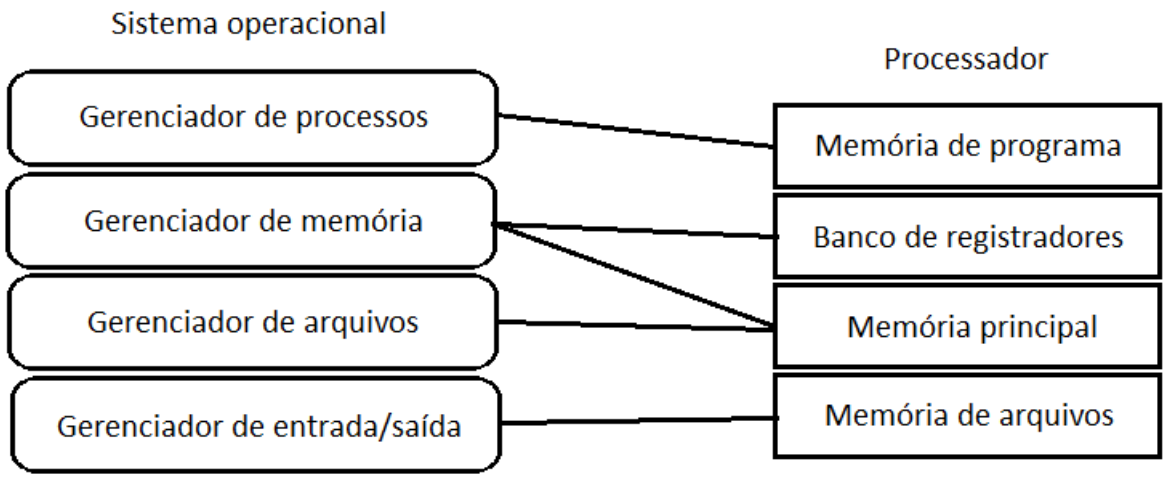
O gerenciamento de entrada e saída será feito por meio de acesso direto à memória, ou seja, dois endereços específicos da memória serão expostos no processador para que possa ser feita a escrita em um deles e a leitura no outro. Para que isso seja possível, será adicionado um novo módulo no processador que faz a exposição dos endereços de forma similar ao que é feito atualmente com as instruções *INPUT* e *OUTPUT*, com a

diferença de que a exposição será de endereços da memória principal, e não de registradores. Além disso, serão criadas duas novas instruções para que seja feita a seleção de quais são endereços serão expostos.

4.5 Interligação entre os blocos

A Figura 8 mostra como é a estrutura de comunicação entre os principais elementos do sistema operacional e os blocos do processador. Como mencionado nas sessões anteriores, o gerenciador de processos gerencia o contador de programa para realizar a preempção e redirecionamento, tendo assim acesso á memória de programa, o gerenciador de memória realiza a movimentação de dados da memória principal e do banco de registradores, o gerenciador de arquivos movimenta os arquivos da memória de arquivos e o gerenciador de entrada/saída realiza a conexão entre os dispositivos externos e endereços específicos da memória principal.

Figura 8 – Interligação entre os blocos do sistema operacional e do processador



Fonte: O Autor

4.6 Alterações no processador

Para integrar os novos conceitos descritos nesse capítulo serão necessárias as seguintes alterações no processador:

- Adicionar um contador de instruções executadas para que seja feita a preempção periódica de programas;
- Adicionar um trecho que realiza a cópia do banco de registradores para um trecho de memória reservado para que as variáveis do programa não sejam perdidas, dado

que o sistema operacional não tem acesso direto ao banco de registradores como tem da memória;

- Adicionar um gerenciador de contador de programa, para que seja feito o redirecionamento entre trechos de código auxiliares, sistema operacional e processos a serem executados;
- Adicionar uma nova memória para armazenar os arquivos gerenciados pelo sistema de arquivos;
- Adicionar módulo de entrada e saída;
- Adicionar duas novas instruções para o módulo de entrada e saída.

4.7 Alterações no compilador

A única alteração que precisará ser feita no compilador é modificar o endereço de memória inicial de acesso dos programas, para que seja sempre acessado o trecho reservado para o processo em execução, mencionado na seção sobre gerenciamento de memória, e os endereços iniciais continuem reservados para o sistema operacional.

5 Considerações Finais

O objetivo do curso é desenvolver um sistema operacional baseado na arquitetura projetada, que seja capaz de realizar as operações básicas esperadas de um sistema operacional, incluindo tudo que for necessário para contemplar, de uma forma geral, gerenciamento de processos, memória, arquivos e entrada/saída. Até o momento, o que este relatório apresenta é apenas o projeto de tudo que será desenvolvido nas próximas etapas, sendo este muito importante para fazer um planejamento consistente.

Referências

1 CLEMENTS, A. *The Principles of Computer Hardware*. 4th edition. ed. Oxford University Press, Inc., New York, NY: [s.n.], 2000. Citado na página 5.

2 CANALTI. *Arquitetura de Computadores (O que é, por que estudar)*. [S.l.], Acessado em 31/03/2019. Disponível em: <https://www.canalti.com.br/arquitetura-de-computadores/arquitetura-de-computadores-o-que-e-por-que-estudar/>. Citado na página 7.

3 GALDINO, D. J. *Aula 03 Organização de Computadores - Processadores - Introdução*. [S.l.], Acessado em 06/04/2019. Citado na página 7.

4 FILHO, R. de G. N. *Fundamentos de Hardware*. [S.l.], Acessado em 06/04/2019. Disponível em: <http://www.di.ufpb.br/raimundo/ArqDI/Arq2.htm>. Citado na página 7.

5 CANALTI. *Arquitetura e Organização de Computadores*. [S.l.], Acessado em 06/04/2019. Disponível em: <https://cadernogeek.wordpress.com/tag/conjunto-de-instrucoes/>. Citado 2 vezes nas páginas 7 e 8.

6 STALLINGS, W. *Computer Organization and Architecture*. 8th edition. ed. Upper Saddle River, NJ 07458: [s.n.], 2009. Citado na página 8.

7 INTEL QUARTUS PRIME DESIGN SOFTWARE. Breaking the Barriers of FPGA Design. Intel FPGA Development Tools. [S.l.], Acessado em 04/05/2019. Disponível em: <https://www.intel.com.br/content/www/br/pt/software/programmable/quartus-prime/overview.html?_ga=2.90972121.25176057.1544534037-954243754.1542224912&erpm_id=7000079>. Citado na página 8.

8 MIDORIKAWA, E. T. *Uma Introdução às Linguagens de Descrição de Hardware*. São Paulo, 2007. Citado na página 9.

9 FPGA. Altera DE2-115 Development and Education Board. 2010. [S.l.], Acessado em 04/05/2019. Disponível em: <https://www.ee.ryerson.ca/~courses/coe608/labs/DE2_115_User_Manual.pdf>. Citado na página 9.

10 [S.l.], Acessado em 16/01/2021. Disponível em: <https://deinfo.uepg.br/~alunoso/2016/ROUNDROBIN>. Citado na página 10.

11 [S.l.], Acessado em 16/01/2021. Disponível em: <https://www.techopedia.com/definition/2767/direct-memory-access-dma>. Citado na página 10.