ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

PCS3732 - Laboratório de Processadores

Mariana Dutra Diniz Costa	12550841
Pedro Henrique Marinho Bressan	12551540
Ygor Acacio Maria	12553688

Engenharia de Computação

Prof. Dr. Bruno Abrantes Basseto

Relatório de Emulação do PDP-7

Implementação em C do histórico minicomputador

São Paulo (SP), 14 de agosto de 2024

Conteúdo

1 Contexto

O PDP-7 (*Programmed Data Processor*) é um minicomputador[1] produzido pela *Digital Equipment Corporation* (DEC) em 1965 [1]. Custando US$ 72.000 na época, mostrou-se uma alternativa mais barata - mas com bom desempenho - aos *mainframes* que custavam 10 vezes mais. Nesse cenário, tendo vendido um total de 120 unidades, o PDP-7 é considerado um sucesso moderado da DEC, o qual abriu portas às gerações seguintes da família PDP que tiveram ainda maior sucesso.

Figura 1: Fotografia de um PDP-7 em restauração em Oslo, Noruega.

Atualmente, restam cinco unidades do PDP-7 incluindo museus e coleções particulares. A figura (1) mostra um PDP-7 em restauração em um museu em Oslo, Noruega.

1.1 Características Principais

Conforme informações extraídas dos manuais da DEC [2, 3], o PDP-7 consiste em um computador digital de propósito geral voltado a aplicações científicas, computacionais ou de controle em tempo real de sistemas da época.

	Consumo	Tensão	Tempo de *Clock*	*Performance*	Preço
PDP-7	2.000 W	115 V, 60 Hz	1,75 μs	0,142 MIPS	US$ 72.000
CDC 6600	30.000 W	208 V, 400 Hz	0,1 μs	2,1 MIPS	US$ 2.370.000

Tabela 1: Comparativo das especificações entre o PDP-7 e o CDC 6600, o mais capaz *mainframe* de 1965.

A tabela (1) mostra algumas das características gerais do sistema PDP-7 se comparado a um *mainframe*. Nota-se que o PDP-7 possui um custo muito menor e possui mais simples integração a redes elétricas convencionais. Evidentemente, o *mainframe* CDC 6600 possui um desempenho muito superior.

Uma das características que permitiram o valor mais barato do PDP-7 era a versatilidade de sua configuração. Como mostra a figura (2), o PDP-7 podia ser aprimorado para diferentes aplicações por meio de módulos de extensão e configurações de periféricos, a exemplo:

- Memória: O *core memory module* de 4096 palavras pode ser estendido a 8192 palavras. Ainda, o uso do controlador externo *memory extension control* é capaz de providenciar endereçamento estendido a 32768 palavras;
- Aritmética: o *extended arithmetic element* é uma opção de extensão do PDP-7 com um módulo capaz de acelerar operações de multiplicação e divisão operando assincronamente;
- Periféricos: por padrão, o PDP-7 somente inclui um simples *teleprinter* e a leitura e escrita de fita perfurada. De acordo com as necessidades do comprador, o computador poderia ser acrescido de

[1] A despeito do nome, "minicomputadores" da época eram do tamanho de armários e pesavam até uma tonelada.

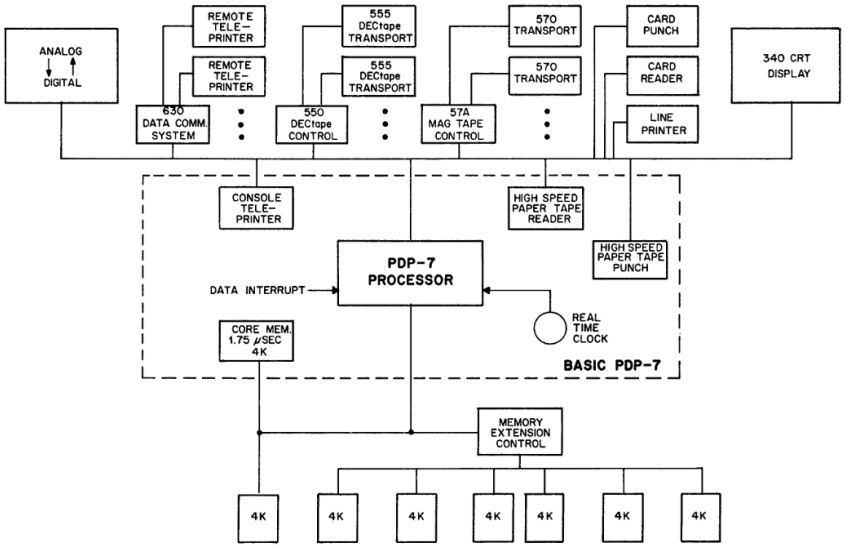

Figura 2: Diagrama do PDP-7 com módulos de expansão. Fonte: [2].

diversas unidades, como operação em cartões perfurados e fitas magnéticas, um *display* gráfico e conversores analógico-digitais.

1.2 Processador (CPU)

A CPU do PDP-7 opera com palavras de 18-bits que opera na forma de um processador acumulador. Por isso, o sistema de numeração de base 8 - octal - é largamente usado na descrição de endereços e valores como abreviação do binário, tendo em vista que o tamanho da palavra é divisível por três (cada dígito octal representa três dígitos binários). Seus registradores estão descritos a seguir:

- *Accumulator* (AC): registrador de 18 bits responsável por guardar operandos e resultados de operações lógicas, além de ser um intermediário para a comunicação com alguns periféricos;
- *Memory Address* (MA): registrador de 13 bits que armazena o endereço da memória em uso;
- *Memory Buffer* (MB): registrador de 18 bits que age como um *buffer* para o valor da memória em uso;
- *Instruction Register* (IR): registrador de 4 bits que armazena o código de operação da instrução a ser executada;
- *Program Counter* (PC): registrador de 13 bits que armazena o endereço da instrução seguinte a ser realizada;
- *Link*: 1 bit auxiliar em operações aritméticas (como indicação de *overflows*).

A figura (3) mostra tais registradores e seus barramentos na CPU.

1.2.1 Conjunto de Instruções (ISA)

A ISA do PDP-7 é composta por instruções de 18 bits de comprimento. A sua decodificação ocorre conforme o seu tipo.

Instruções de Memória: As instruções de memória recebem esse nome, pois o valor do operando para tais instruções é lido da memória conforme o endereço descrito na instrução. Essas instruções seguem o formado regular a seguir, conforme ilustrado pela figura (4):

1. OpCode (bits 0 a 3): identificador de cada instrução de memória, salvo no IR;
2. Bit de Indireção (bit 4): bit utilizado para indicar se o endereço do operando deve ser usado diretamente ou lido como um ponteiro a um outro endereço;

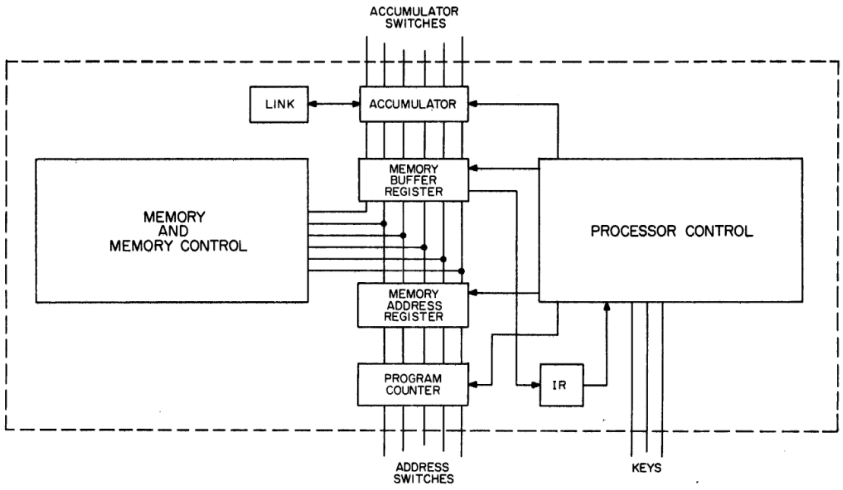

Figura 3: Diagrama da CPU mostrando seus principais registradores. Fonte: [2].

3. Endereço do Operando (bits 5 a 17): endereço do operando a ser utilizado na instrução, esse endereço e o valor contido nele são salvos no MA e MB respectivamente.

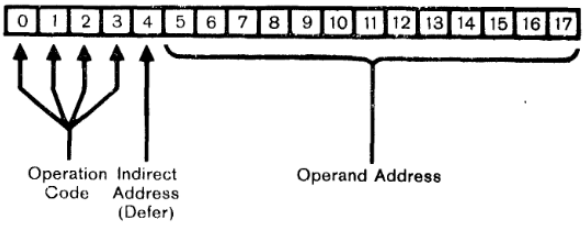

Figura 4: Decodificação dos bits das instruções de memória. Fonte: [2].

As instruções de memória estão detalhadas na tabela (2).

Mnemônico	Código Octal	Operação
`lac Y`	20	Carrega o AC com o conteúdo da memória Y.
`dac Y`	04	Deposita o AC na memória Y.
`dzm Y`	14	Deposita zero na memória Y.
`add Y`	30	Soma o conteúdo de Y ao AC em complemento de 1.
`tad Y`	34	Soma o conteúdo de Y ao AC em complemento de 2.
`xor Y`	24	Realiza a operação XOR entre Y e o AC.
`and Y`	50	Realiza a operação AND entre Y e o AC.
`jmp Y`	60	Salta para a instrução na memória Y.
`jms Y`	10	Salta para sub-rotina em Y.
`cal`	00	Chama sub-rotina (equivalente a jms 20).
`xct Y`	40	Executa a instrução na memória Y.
`sad Y`	54	Ignora se AC for diferente de Y.
`isz Y`	44	Incrementa Y e salta se o resultado for zero.

Tabela 2: Códigos de operação e descrição das instruções de memória do PDP-7.

Instruções de Operação: As instruções de operação abrangem as operações sobre os registradores do PDP-7, como o AC e o Link, que não necessitam de um operando da memória. Ainda, também podem realizar execuções condicionais sobre a instrução seguinte a depender do valor dos registradores.

As instruções de operação estão detalhadas na tabela (3).

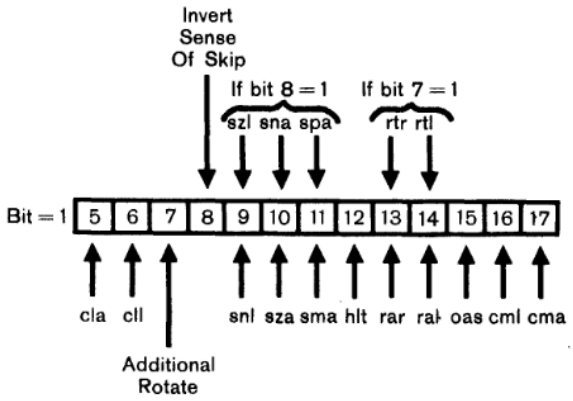

Figura 5: Decodificação dos bits das instruções de operação. Fonte: [2].

Mnemônico	Código Octal	Operação
`opr`	740000	Executa instruções de operação baseadas nos bits.
`cla`	750000	Limpa o AC.
`cma`	740001	Complementa o AC.
`cll`	744000	Limpa o Link.
`cml`	740002	Complementa o Link.
`ral`	740010	Rotaciona o AC e o Link à esquerda.
`rtl`	742010	Rotaciona duas vezes o AC e o Link à esquerda.
`rar`	740020	Rotaciona o AC e o Link à direita.
`rtr`	742020	Rotaciona duas vezes o AC e o Link à direita.
`oas`	740004	Realiza o *or* entre os *switches* e AC.
`sma`	740100	Ignora se o AC for negativo.
`spa`	741100	Ignora se o AC for positivo.
`sza`	740200	Ignora se o AC for zero.
`sna`	741200	Ignora se o AC não for zero.
`snl`	740400	Ignora se o Link não for zero.
`szl`	741400	Ignora se o Link for zero.
`htl`	740040	Interrompe a operação do computador.

Tabela 3: Código de operação e descrição das instruções de operação.

1.3 *Type 340 Precision Incremental Display*

O PDP-7 poderia ser opcionalmente acompanhado do monitor *Type 340 Display*. A tela é de tecnologia CRT e possui resolução de 1024x1024 pontos [4]. Essa tela CRT integra-se como periférico ao I/O do PDP-7 e permite a visualização desde linhas simples e caracteres até gráficos vetorizados. Um exemplo de uso do monitor está em um dos primeiros *video-games*: *Space Travel* desevolvido por Ken Thompson na Bell Labs [5].

O monitor possui *frame* circular, conforme mostra a figura (6).

1.3.1 Modos de Operação

Conforme indicado no manual de operação do monitor [4], este aceita instruções de 18 bits e possui diferentes modos de operação, dependendo do que se quer exibir. Os principais modos de operação estão descritos a seguir, a lista completa encontra-se em [4]:

- *Parameter Mode*: modo ocioso do monitor, em que ele aguarda uma instrução para alterar a outro modo;
- *Point Mode*: modo que recebe instruções contendo coordenadas x e y e exibe um ponto nesse local;
- *Vector Mode*: recebe instruções com valores Δx e Δy de modo a definir uma magnitude e direção a se exibir um segmento na tela;

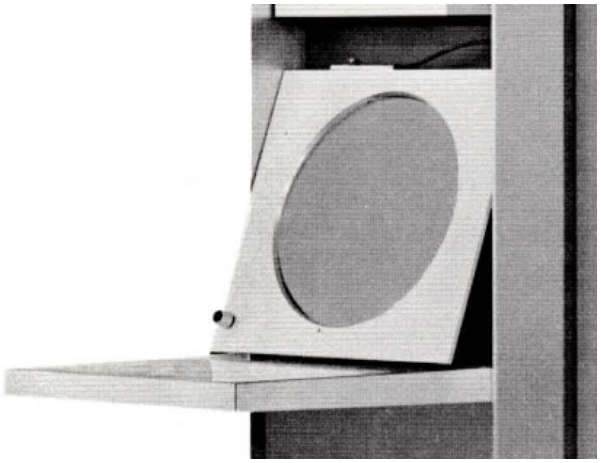

Figura 6: Trecho de fotografia do PDP-7 destacando seu *display*. Fonte: [4].

- *Character Mode*: se equipado da extensão *Type 342 Character Generation*, o modo caractere permite a exibição de três caracteres por instrução.

1.4 Contexto Histórico

Apesar do PDP-7 ter vendido um número de unidades que, décadas depois, seria uma fração ínfima dos computadores vendidos diariamente, sua importância histórica a diversos aspectos da computação permaneceria.

1.4.1 Sistema Operacional Unix

Desenvolvido na Bell Labs por Ken Thompson, Dennis Ritchie e outros, os primeiros protótipos do Unix (na época nomeado Unics) foram programados em *assembly* no PDP-7 como sistema operacional para o *video-game Space Travel*. Diversas decisões de projeto do sistema operacional advieram de características da família PDP, como o uso eficiente do espaço de endereçamento e o não compartilhamento de memória entre processos [6]. Os Sistemas Unix influenciaram fortemente o desenvolvimento do Linux e MacOS, dois dos maiores sistemas operacionais utilizados atualmente.

1.4.2 Linguagem de Programação B

Elaborada por Ken Thompson e Dennis Ritchie na Bell Labs, a linguagem B, ainda que praticamente extinta atualmente, foi um marco na computação por ser precursora da linguagem C, uma das mais utilizadas e influentes na história da computação. Diversas características do PDP-7 foram determinantes ao desenvolvimento do B, uma linguagem sem tipos, tendo em vista que o PDP-7 tem somente palavras de 18 bits; outra característica ubíqua na computação atual é a presença de ponteiros, que se deve ao bit de indireção da ISA da família PDP.

```
1    printn(n, b) {
2        extrn putchar;
3        auto a;
4
5        if (a = n / b)        /* assignment, not test for equality */
6                printn(a, b);
7        putchar(n % b + '0');
8    }
```

Código 1: Exemplo de trecho de código B. [7]

1.5 Emulação

O presente projeto refere-se à emulação do PDP-7 na linguagem C. O presente relatório elenca a motivação para o projeto, bem como seus objetivos, modelo de problema e solução e o protótipo realizado. Nesse sentido, projetos anteriores, como o *Computer History Simulation Project* [8], são de grande influência.

2 Motivação

A motivação para o projeto de emulação do PDP-7 tem motivo principalmente acadêmico e preservacionista.

No sentido educacional, o projeto de emulação é essencial ao entendimento do funcionamento de processadores no início da computação e as soluções em *hardware* e *software* implementadas que por vezes não tiveram sucesso e outras permanecem na atualidade. Por exemplo, apesar da simplicidade de seu conjunto de instruções, o funcionamento do PDP-7 e os programas nele escritos tornam claras decisões de projeto que permanecem relevantes atualmente - como o bit de indireção - já outros fatores, como as palavras de 18 bits, atualmente foram abandonadas nos processadores comuns de *desktop* ou embarcados ARM de 16, 32 ou 64 bits.

Outrossim, a motivação de preservação é outra característica importante do projeto ao dar maior destaque a processadores frequentemente negligenciados hoje (por exemplo por seu tamanho de palavra exótico e relativo baixo número de vendas) como malsucedidos, quando - na prática - tiveram relevante papel em várias áreas do desenvolvimento da computação.

3 Objetivos

Dada a contextualização e a importância histórica do PDP-7, foi decidido desenvolver o emulador Podepaset em linguagem C, capaz de executar programas escritos em código de máquina conforme a ISA do processador. Nesse sentido, foram definidos os seguintes objetivos para o projeto:

- Implementar a arquitetura de conjunto de instruções do processador PDP-7 em linguagem C.
- Realizar a integração com periféricos, como monitor, para proporcionar uma interface análoga ao do operador.
- Executar corretamente de programas escritos em código de máquina e carregados na memória.
- Analisar o desempenho do PDP-7 com o número de ciclos necessário à execução de programas.
- Demonstrar na prática a execução de programas simples para validar a funcionalidade do emulador completo desenvolvido.

4 Modelo do Problema

O problema consiste em como realizar a emulação da CPU e a integração ao *display* dado, somente, o código de máquina do programa a ser executado e o estado inicial da memória. Nesse sentido, é necessário computar os valores de cada registrador da CPU a cada operação. Os seguintes fatores são entradas do problema pensadas no início do desenvolvimento:

- Código de máquina do programa e memória. De forma análoga à configuração de uma fita perfurada. A exemplo:

  `000000 000001` ; número 1 na primeira posição da memória

- Exemplo de saída desejada da emulação:

  `IR = 0; PC = 1; AC = 0; MA = 1` ; Ciclos Executados: 4

5 Modelo de Solução

A solução para o problema apresentado foi planejada de forma a receber do usuário os arquivos de programa de memória do PDP-7; então o emulador é responsável por executar as etapas de *fetch*, *decode* e *execute* de cada instrução. Dessa forma, em cada uma dessas etapas são atualizados os registradores da CPU de modo a se obterem as saídas desejadas.

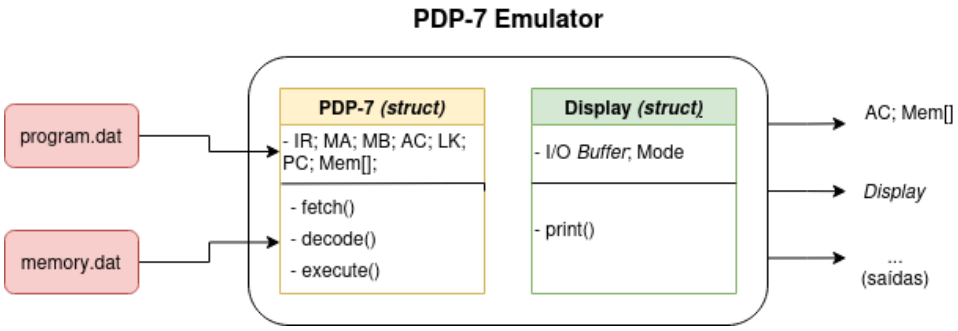

Figura 7: Diagrama do modelo de solução da emulação.

6 Protótipo

6.1 Implementação

A implementação do Modelo de Solução se deu por meio das definições das figuras (8, 9).
O módulo `pdp7_cpu.c` possui três funções principais para simular cada ciclo da CPU:

- *fetch*: realiza o resgate da próxima instrução e incrementa o PC;
- *decode*: faz o *parsing* da instrução a ser executada por meio de operações de *bit-shift* de modo a se obter cada campo das figuras (4, 5);
- *execute*: possui um `switch` que selecionada dentre todas as operações disponíveis aquela correta para a instrução decodificada.

De forma semelhante, o módulo `display.c` aguarda até que o seu *buffer* seja preenchido e escreve no componente gráfico `SDL_Renderer` o que está contido no *buffer*.

```c
typedef struct {
    uint32_t accumulator;           // Accumulator (18-bit)
    uint32_t memory_address;        // Memory Address (13-bit)
    uint32_t memory_buffer;         // Memory Buffer (18-bit)
    uint32_t memory[MEMORY_SIZE];   // Memory array (18-bit words)
    uint32_t pc;                    // Program Counter (13-bit)
    uint32_t* io_buffer;            // I/O Buffer addr
    uint8_t ir;                     // Instruction Register (4-bit)
    bool link;                      // Link Register (1-bit)
    uint64_t cycles;                // Cycle counter
    bool running;                   // CPU running state
} PDP7_cpu;
```

Figura 8: *Struct* C utilizada para definir a CPU.

```c
typedef struct {
    int mode;               // Current display mode
    bool running;           // Flag if display is running
    uint32_t* io_buffer;    // Pointer to the IO buffer
    SDL_Renderer* renderer; // SDL Window Renderer
} display_340;
```

Figura 9: *Struct* C utilizada para definir o *display*.

A implementação detalhada encontra-se disponível no GitHub [9].

6.2 Integração

A CPU e o *display* foram estruturados em *threads* separados de forma a simular, de fato, o funcionamento independente desses dois componentes. A integração e sincronismo entre eles foi feita por meio

de uma variável compartilhada denominada `io_buffer`, a qual é uma versão análoga simplificada ao funcionamento DMA desse canal do *display* no PDP-7.

6.3 Teste

A fim de testar a implementação do emulador Podepaset, foi escrito um programa de Fibonacci utilizando o conjunto de instruções originais do PDP-7. Esse programa, apresentado nos Códigos 2 e 3, calcula os N primeiros valores da sequência de Fibonacci, para um N previamente definido na posição de memória 0, e os armazena em posições de memória contíguas iniciadas no endereço 20. Além disso, cada valor calculado é impresso no display do usuário, por meio da implementação da instrução de E/S TLS.

```
00000 000034 // N - espaco pra input do usuario
00001 000000 // FIB1 - primeiro numero da sequencia
00002 000001 // FIB2 - segundo numero da sequencia
00003 000000 // TEMP - reserva temporaria para adicao
00004 000000 // COUNT- contador de loop
00005 000000 // NEXT - proximo numero da sequencia (fib1 + fib2)
00006 000022 // SEQ_LOC - endereco atual pra guardar o next (atualizado a cada iteracao)
00007 777776 // -2 - constante
00010 777777 // -1 - constante

00020 000000 // SEQ  - endereco de inicio pra guardar a sequencia de fibonacci calculada
00021 000000 // SEQ+1
```

Código 2: Memória inicial para o código de Fibonacci.

```
// START
00000 750000 // CLA      - clear AC
00001 750000 // CLA      - clear AC - nop
00002 200000 // LAC 0    - AC<-N
00003 741200 // SNA      - pula se AC != 0
00004 600100 // JMP END  - se AC==0, termina o programa (jump para END)
00005 200001 // LAC 1    - AC<-FIB1 (0)
00006 040020 // DAC 20   - SEQ<-FIB1
00007 200002 // LAC 2    - AC<-FIB2
00010 040021 // DAC 21   - SEQ+1<-FIB2
00011 200000 // LAC 0    - AC<-N
00012 340007 // TAD 7    - AC<-N-2
00013 040004 // DAC 4    - COUNT<-N-2
// LOOP
00014 200004 // LAC 4         - AC<-COUNT
00015 741200 // SNA           - pula se COUNT != 0
00016 600100 // JMP END       - se COUNT==0, termina o programa
00017 600031 // JMP FIB_CALC  - calcula o proximo numero da sequencia - subrotina
00020 200004 // LAC 4         - AC<-COUNT
00021 340010 // TAD 10        - AC<-COUNT-1
00022 040004 // DAC 4         - COUNT<-COUNT-1
00023 600014 // JMP LOOP      - repete loop

// FIB_CALC - subrotina para calcular o proximo fibonacci; atualizar os valores de fib1,
    fib2, next, seq_loc; e mostrar o valor calculado no display
00030 750000 // CLA      - clear AC
00031 200002 // LAC 2    - AC<-FIB2
00032 040003 // DAC 3    - TEMP<-FIB2
00033 340001 // TAD 1    - AC<-AC+FIB1
00034 040005 // DAC 5    - NEXT<-FIB2+FIB1
00035 200003 // LAC 3    - AC<-TEMP
00036 040001 // DAC 1    - FIB1<-TEMP
00037 200005 // LAC 5    - AC<-NEXT
00040 040002 // DAC 2    - FIB2<-NEXT
00041 060006 // DAC I 6  - [SEQ_LOC]<-NEXT
00042 440006 // ISZ 6    - SEQ_LOC<-SEQ_LOC+1
00043 200005 // LAC 5    - AC<-NEXT
00044 700406 // TLS      - imprime AC no display
00045 600020 // JMP 20   - retorna da subrotina

// END - halt
00100 740040
```

Código 3: Código de máquina (ISA PDP-7) para calcular sequência de Fibonacci.

Com isso, o código de máquina Fibonacci foi carregado nos respectivos arquivos *memory* e *program*, o que permitiu executá-lo no emulador. Para esse teste, N foi escolhido como 34 (octal), que corresponde ao 28° número da sequência de Fibonacci iniciada em zero e leva o valor de 577502 (octal) ou 196418 (decimal). Essa escolha foi feita pois e é o maior valor da sequência capaz de ser calculado dentro das limitações de 18 bits do PDP-7.

Por meio do comando *"make debug"*, foi possível rodar o programa e acompanhar os valores de depuração do contador de programa, acumulador, endereço de memória, instrução e contagem de ciclos. Dessa forma, após 939 ciclos, o programa foi concluído, contando com a abertura da janela de *display* e a impressão da sequência de Fibonacci. Na Figura 10, se observa a saída gerada pelo programa em octal e os valores correspondentes em decimal para conferência.

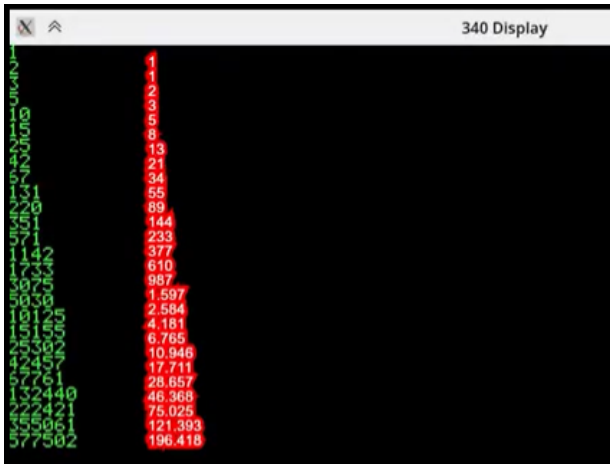

Figura 10: Saída do programa de Fibonacci no *display*.

Com a interface de depuração desenvolvida, também foi possível verificar os resultados desejados na memória do processador emulado. Além de acompanhar os valores de PC, AC, MA, IR e Ciclos, depois do *halt* foi inspecionada a memória. Assim, como mostra a Figura 11, a sequência de Fibonacci calculada foi vista corretamente armazenada a partir do endereço 20, conforme esperado.

```
PC:   2043 | AC: 577502 | MA:      6 | IR:     44 | Cycles:    921
PC:   2044 | AC: 577502 | MA:      5 | IR:     20 | Cycles:    923
PC:   2045 | AC: 577502 | MA:    406 | IR:     70 | Cycles:    923
PC:   2020 | AC: 577502 | MA:     20 | IR:     60 | Cycles:    924
PC:   2021 | AC:      1 | MA:      4 | IR:     20 | Cycles:    926
PC:   2022 | AC:      0 | MA:     10 | IR:     34 | Cycles:    928
PC:   2023 | AC:      0 | MA:      4 | IR:      4 | Cycles:    930
PC:   2014 | AC:      0 | MA:     14 | IR:     60 | Cycles:    931
PC:   2015 | AC:      0 | MA:      4 | IR:     20 | Cycles:    933
PC:   2016 | AC:      0 | MA:   1200 | IR:     74 | Cycles:    934
PC:   2100 | AC:      0 | MA:    100 | IR:     60 | Cycles:    935
PC:   2101 | AC:      0 | MA:     40 | IR:     74 | Cycles:    939
CPU halted
Total cycles: 939
Commands: [m]emory, [q]uit
> m
Enter start address: 0
Enter end address: 60
Memory contents from 0000 to 0060:
0000: 000034 355061 577502 355061 000000 577502 000054 777776
0010: 777777 000000 000000 000000 000000 000000 000000 000000
0020: 000000 000001 000001 000002 000003 000005 000010 000015
0030: 000025 000042 000067 000131 000220 000351 000571 001142
0040: 001733 003075 005030 010125 015155 025302 042457 067761
0050: 132440 222421 355061 577502 000000 000000 000000 000000
0060: 000000
```

Figura 11: Saída do programa de Fibonacci no terminal.

7 Considerações Finais

À luz das informações e resultados apresentados, fica evidente que o projeto de emulação do PDP-7 decerto atendeu às motivações acadêmicas e históricas que suscitaram o interesse. Dentro do contexto

discutido na Seção 1, foi de grande relevância a pesquisa histórica por esse processador do início da computação moderna. Não somente, a aparente simplicidade do funcionamento do PDP-7 foi sobrepujada pela relevância dele, e de seus modelos subsequentes, a vários aspectos da computação hoje. Deve-se notar, sem dúvida, o hercúleo trabalho das mentes de Ken Thompson e Dennis Ritchie, dentre outros, nos primeiros desenvolvimentos realizados no PDP-7 pela Bell Labs do Sistema Unix e a linguagem B que se tornariam mais tarde precursores dos mais utilizados sistemas operacionais e linguagens de programação do mundo.

Os resultados obtidos após os testes indicam que os objetivos definidos para o projeto foram alcançados, incluindo a implementação das instruções da ISA em C, a integração com o *display* de usuário e as possibilidades de depuração e análise de desempenho dos programas testados. Percebe-se, por fim, além de um exercício técnico valioso, o emulador como uma ferramenta eficaz para a exploração educacional e técnica, tanto da própria arquitetura do PDP-7 quanto do funcionamento de processadores em geral.

Referências técnicas

[1] *Computers and Automation Newsletter*. XIII. Último Acesso: 2024-08-14. 1964, p. 45. URL: http://www.bitsavers.org/magazines/Computers_And_Automation/196407.pdf.

[2] *PDP-7 Reference Manual*. Último Acesso: 2024-08-14. URL: http://bitsavers.trailing-edge.com/pdf/dec/pdp7/F-75P_PDP7prelimUM_Dec64.pdf.

[3] *PDP-7 Maintenance Manual*. Último Acesso: 2024-08-14. URL: http://bitsavers.informatik.uni-stuttgart.de/pdf/dec/pdp7/F-77A_pdp7maint_1966.pdf.

[4] *Type 340 Precision Display Manual*. Último Acesso: 2024-08-15. URL: http://www.bitsavers.org/pdf/dec/graphics/H-340_Type_340_Precision_Incremental_CRT_System_Nov64.pdf.

[5] Dennis Ritchie. *Space Travel: Exploring the solar system and the PDP-7*. Último Acesso: 2024-08-14. 2001. URL: https://www.bell-labs.com/usr/dmr/www/spacetravel.html.

[6] Dennis Ritchie. *The UNIX Time-sharing System - A Retrospective*. Último Acesso: 2024-08-14. 1977. URL: https://www.bell-labs.com/usr/dmr/www/retro.pdf.

[7] K Thompson. *Users' Reference to B*. Último Acesso: 2024-08-15. 1972. URL: https://web.archive.org/web/20150611114427/https://www.bell-labs.com/usr/dmr/www/kbman.pdf.

[8] *The Computer History Simulation Project - GitHub Repository*. Último Acesso: 2024-08-14. URL: https://github.com/simh/simh.

[9] *Podepasete - GitHub Repository*. Último Acesso: 2024-08-14. URL: https://github.com/phmbressan/podepasete.