Relatório
Avaliação da Calculadora de 16 bits

Autor: Jonas Bezerra da Costa Máximo
Professora: Roberto Cabral Rabêlo Filho
Curso: Engenharia de Computação

1. **Introdução:**
   O presente relatório aborda o desenvolvimento de uma calculadora de 16 bits em assembly ARM na cadeira de arquitetura e organização de computadores 2, na qual foi o utilizando a placa Beaglebone Black com o sistema de programação Bare Metal que utilizou o cross compilado arm-linux-gnueabihf para compilar o código que foi enviado para a placa através do cabo ethernet utilizando o protocolo TFTP.

   A calculadora funciona da seguinte forma: (valor) espaço (valor) espaço (operando) o resultado será mostrado nos 4 LEDs da placa

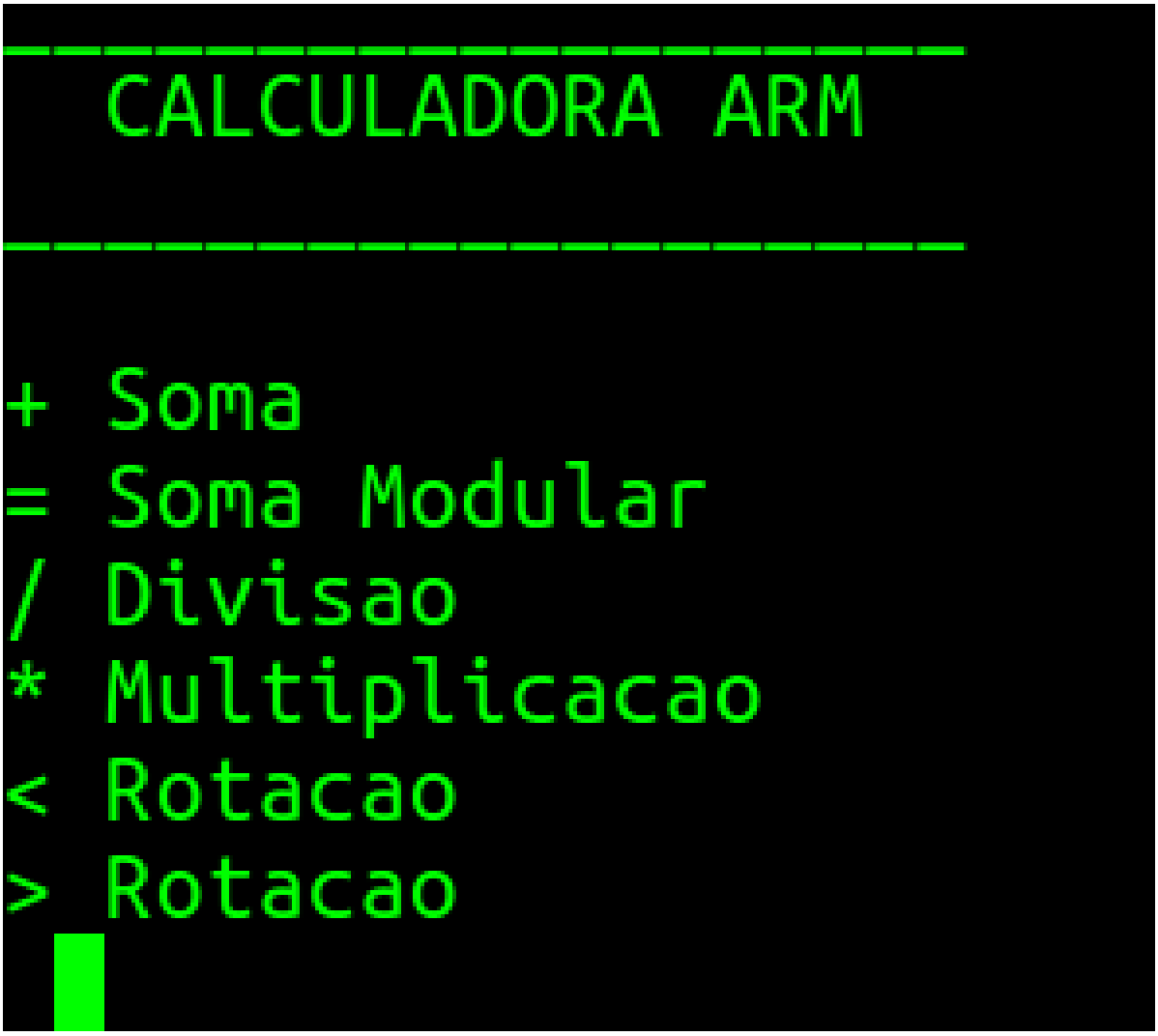

Figura 1: menu de ultilização da calculadora

2. **Defines e Inicializações**
   Na primeira parte do código temos alguns defines e inicializações como do Clock, Watch Dog, GPIO,

UART, LEDs e botão no qual o valores foram consultados no detashet e no esquemático [1].

Inicialmente encontrei um pouco de dificuldade por causa da quantidade de informações presentes no arquivo de consulta.

3. **Main**
   Ao iniciarmos a main temos algumas chamadas de strings que compõem o menu principal, temos o GET no qual receberá uma entrada do usuário onde será submetida a algumas verificações, se o mesmo é um número ou operando, qualquer valor ou operando diferente do esperado resultará em um erro fatal no qual os 4 LEDs da placa piscarão 5 vezes e a calculadora será reinicializada e voltará ao menu inicial.

   Caso o valor seja válido ele printará o valor e entrará em um loop que faz a multiplicação das entradas para que possam ter mais de um algarismo, caso seja pressionado espaço irá receber o próximo número que utiliza o mesmo processo de ampliação de algarismos, após o entre ser pressionado novamente fará a recepção do operando em questão no qual chamará suas devidas funções e seus retornos setarão os devidos LEDs.

   Na main encontrei imensa dificuldade em trabalhar com strings pois as mesmas geravam um erro de constante inválida muito provavelmente ocasionada pela alocação dessa string na memória e o erro não seguia um padrão de funcionamento no qual se adicionássemos ou retirássemos um espaço o mesmo voltava a funcionar, não tive mais dificuldades com o GET pois foi o mesmo código utilizado em práticas passadas.

4. **+ Soma**
   Na labal soma recebo os dois valores da main e os somo usando ADD após a soma comparamos se o resultado e maior que 15 se for ele gerará um erro no qual piscará os LEDs 0 e 3 3 vezes com um tempo de cerca de 1s entre acender e apagar, se não imprimirá normalmente o valor nos LEDs

   Não houve dificuldade na implementação dessa função.

5. **= Soma Modular**
   Na labal soma modular recebo os dois valores da main os somo usando o ADD e entro em um loop no qual eu os comparo com 16 se o mesmo for menor eu retorno para a main e os imprimo normalmente no LEDs caso contrário subtraio 16 e os comparo novamente atáé ser menor que 16 não há erros nessa label.

   Não houve dificuldade na implementação dessa função.

6. **/ Divisão**
   Na labal divisão recebo os dois valores da main o divisor e o dividendo, inicialmente comparo o divisor com zero se o mesmo for igual aciona a lebal de erro da divisão que pisca os LEDs 1 e 2 3 vezes e retorna para a main, no caso de o divisor ser maior que zero mas o resultado ser maior que 15 ocorrerá um erro fatal no qual piscará os 4 LEDs 5 vezes e retornará para a main, se o mesmo não entar em nenhum caso dos anteriores retornará para a main e printará os leds normalmente.

   Para o desemvolvimento dessa label foi consultado os slides disponibilisados pelo professor e adaptado o código de divisão[2]

7. *** Multiplicação Modular**
   Na labal multiplicação modular recebo os dois valores da main e início o primeiro loop que somara o primeiro valor com ele mesmo a quantidade de vezes do segundo número que irá sendo decrementado até zero, quando o mesmo for zero sairá do loop e entrará em um segundo loop o qual fará a modularização, ele comparará se o resultado da multiplicação é maior ou igual a 16 enquanto for ele decrementará 16 desse valor e o que sobrar será o retorno da labal que será printada na main

Nessa labal foi utilizada novamente o código da soma adaptada disponibilizada nos slides do professor, houve uma pequena dificuldade na tentativa de utilizar o bx ir a função de divisão já implementada anteriormente visto que já havia utilizado um bx lr para estar na multiplicação modular

8. < **Rotação para a esquerda**
   Na labal rotação para a esquerda recebo os dois valores da main, o valor inicial e o valor de rotação, caso o valor inicial for maior que 15 vai gerar o erro fatal no qual piscará os 4 LEDs da placa 5 vezes, caso seja menor o valor inicial iniciará o loop onde será deslocado 1 vez para a esquerda após isso será feito um cópia e deslocado para a direita 4 vezes logo após será feiro uma ORR e entre esses registradores e uma AND com 15 para zerar os bits restantes e decremento o valor de rotação, quando o mesmo for zero sairá do loop e retornará para a mian onde imprimirá o novo valor rotacionado

   Nessa label encontrei dificuldade na lógica onde pedi ajuda napa o aluno Daniel Mascarenhas onde o mesmo me explicou a lógica dele e eu implementei a mesma lógica

9. > **Rotação para a direita**
   Na labal rotação para a direita recebo os dois valores da main, o valor inicial e o valor de rotação, caso o valor inicial for maior que 15 vai gerar o erro fatal no qual piscará os 4 LEDs da placa 5 vezes, caso seja menor o valor inicial iniciará o loop onde será feito uma rotação ROR 1 vez logo após serpa feiro uma cópia em outro registrador que será deslocado 28 vezes para a direita e será feiro uma ORR com o valor do registrador e da sua cópia e decremento o valor de rotação, quando o mesmo for zero sairá do loop e retornará para a mian onde imprimirá o novo valor rotacionado

   Nessa label encontrei dificuldade na lógica onde pedi ajuda napa o aluno Daniel Mascarenhas onde o mesmo me explicou a lógica dele e eu implementei a mesma lógica

10. **Botão**
    A label referente ao botão e sempre acionada após cada label de operando ele configura o GPIO base mais o GPIO_DATAIN é setado o 7 que e referente ao GPIO_1 do pino físico 4 do P8, após isso e deslocado 7 apenas para o bit de verificação dicar na primeira posição para podermos comparar com 1, enquanto esse bit não for igual a 1 ele fica no loop quando o botão e acionado sai do loop e os LEDs são apagados e feito um jump incondicio

    Tive dificuldade com a implementação do botão e foi necessário o auxílio dos estudantes Elias e Gabriel no qual me explicaram o funcionamento do GPIO_DATAIN com isso pude fazer o botão funcionar

11. **Concideraçõe Finais**
    Para a realização dessse trabalho foram necesários várias horas de estudo e desemvolvimento mesmo assim não foi possível implementar algumas funcionalidades como o botão funcionar por meio da interrupção e a modularização do código, mas obtemos muito conhecimento nesse.

12. **Referências**
    [1] https://github.com/beagleboard

    [2] slides do professor Roberto Cabral

    [3] https://github.com/allexoll/BBB-BareMetal