Trabalho 2 - Parte 1

Introdução

Subcamada LoCo
- Lógica de controle do programa

Subcamada BiCo

Subcamada SoCa
- Descrição das Syscalls
- Iniciando o Sistema
- Descrição dos Periféricos
  - General Purpose Timer
  - Sensores de Luminosidade
  - Sensor Ultrassônico
  - Global Position System e Giroscópio
- Mapeamento dos Registradores dos Periféricos na Memória
  - Realizando leitura de um registrador mapeado em memória

Infraestrutura

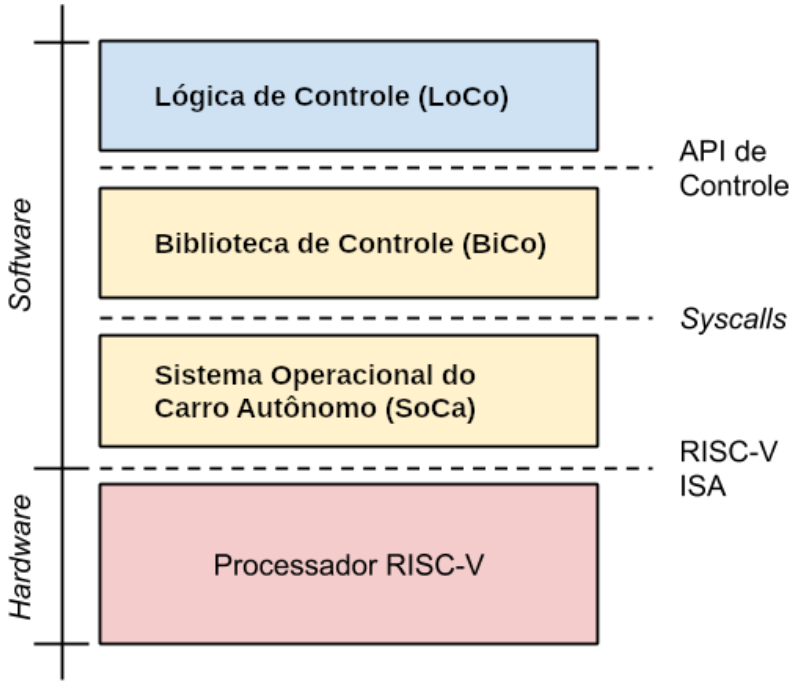
Entrega

Dicas e Observações

Introdução

Carros autônomos são veículos capazes de reconhecer seu ambiente e mover-se seguramente com pouca ou nenhuma intervenção humana. Recentemente, grandes investimentos foram destinados à sua pesquisa visando colocar estes veículos nas ruas e, consequentemente, a empresa *Borinet Motors* foi fundada, atuando neste ramo. Esta empresa contratou você para cuidar do projeto do *self-driving car*, um carro autônomo que consegue percorrer as ruas de uma cidade, sem sair dela ou bater/esbarrar em nenhum objeto durante seu trajeto.

Neste segundo trabalho da disciplina, você desenvolverá as camadas de *software* responsáveis pelo controle do *self-driving car*. Tais camadas, ilustradas na Figura abaixo, são divididas em três subcamadas, a saber: (a) Sistema Operacional do Carro Autônomo (SoCa); (b) Biblioteca de Controle (BiCo) e; (c) Lógica de Controle (LoCo).



A subcamada SoCa é responsável pelo gerenciamento do *hardware*. Esta deve prover um conjunto de serviços para a subcamada BiCo através de chamadas de sistemas, ou *syscalls*. A subcamada SoCa contém código que será executado em modo supervisor e deve ser implementada em linguagem de montagem do RISC-V.

A subcamada BiCo é responsável por prover uma interface de programação amigável para a Lógica de Controle, a API de Controle. A subcamada BiCo também deve ser implementada em linguagem de montagem, mas seu código será executado no modo usuário e ligado com o código da subcamada LoCo com o auxílio do ligador (*linker*).

A subcamada LoCo é responsável pela lógica de controle do carro e deve invocar as funções definidas pela API de Controle, implementadas pela BiCo. A subcamada LoCo deve ser implementada em código na linguagem C e seu código será executado no modo usuário. Como informado acima, o código da LoCo deverá ser ligado ao código da subcamada BiCo com o auxílio do ligador.

O código auxiliar para este trabalho está disponível neste link.

Subcamada LoCo

O código da subcamada LoCo deve ser implementado em linguagem C e deve fazer uso das rotinas disponíveis na API de Controle (arquivo `api_car.h`) para enviar comandos para o carro. Você desenvolverá um programa na linguagem C para a camada LoCo no arquivo o `loco.c`.

Lógica de controle do programa

Nesta primeira parte do trabalho você deverá fazer o carro se mover autonomamente por uma pista até um destino, sem sair da pista ou esbarrar/bater em algo, guiando-se por um sensor de luminosidade, disposto à frente do carro (descrito posteriormente). Desta forma, sua lógica de controle deve utilizar as funções da API de controle de maneira correta para que este objetivo seja alcançado.

Além disso, também para esta primeira parte do trabalho, os pontos de início e destino são fixos e o carro já começa em seu ponto inicial (e rotação inicial) sempre que o dispositivo self_driving_car é adicionado ao simulador. Seu ponto de início é mostrado na figura abaixo:

O carro deve percorrer esta pista até seu final, conforme mostrado na Figura abaixo, em no máximo 480 segundos (8 minutos).

Subcamada BiCo

O código da subcamada BiCo deve implementar as rotinas da API de Controle em linguagem de montagem do RISC-V. A API está descrita no arquivo `api_car.h` e deve ser implementada no arquivo `bico.s`. Para controlar o *hardware*, o código deve realizar chamadas ao sistema (instrução `ecall`), ou *syscalls*. As *syscalls* são definidas na seção conseguinte.

Subcamada SoCa

Você deve programar a subcamada SoCa, no arquivo `soca.s`, para atender as chamadas de sistemas descritas a seguir. Seu tratador de chamadas de sistema deve analisar o valor contido no registrador `a7` para identificar a chamada de sistema solicitada.

Descrição das Syscalls

A tabela abaixo apresenta os dados das syscalls do sistema

Syscall	Parâmetros	Retorno	Descrição
Syscall_set_motor Código: 10	a0: valor para deslocamento vertical a1: valor para o deslocamento horizontal	0 em caso de sucesso e -1 em caso de falha (parâmetros inválidos)	Aciona os motores para o deslocamento vertical e horizontal. O valor de a0 pode ser -1 (andar para trás), 0 ou 1 (andar para frente). O valor de a1 varia de -127 a +127, onde valores negativos geram deslocamentos para a direita e positivos para a esquerda.
Syscall_set_handbreak Código: 11	a0: valor dizendo se o freio deve ser acionado	-	a0 deve conter 1 para acionar o freio de mão.
Sycall_read_sensors Código: 12	a0: endereço de um vetor de 256 elementos que armazenará os valores lidos do sensor de luminosidade	-	Lê os valores do sensor luminosidade
Syscall_read_sensor_distance Código: 13	-	Valor obtido na leitura do sensor; -1 caso nenhum objeto tenha sido detectado a menos de 20 metros	Lê o valor do sensor ultrasônico que detecta objetos até 20 metros a frente do carro
Syscall_get_time	-	Um inteiro sem	Lê o tempo do sistema

Código: 14		sinal contendo tempo do sistema, em milisegundos	
Syscall_get_position Código: 15	a0: endereço da variável que armazenará o valor da posição x a1: endereço da variável que armazenará o valor da posição y a2: endereço da variável que armazenará o valor da posição z	-	Lê a posição aproximada do carro usando o dispositivo de GPS
Syscall_get_rotation Código: 16	a0: endereço da variável que armazenará o valor do ângulo de Euler em x a1: endereço da variável que armazenará o valor do ângulo de Euler em y a2:endereço da variável que armazenará o valor do ângulo de Euler em z	-	Lê a rotação global do dispositivo de giroscópio

Iniciando o Sistema

Ao iniciar o sistema, o SoCa deve realizar transferir a execução para a aplicação de controle no modo usuário. Para tanto, o SoCa deve:

- Configurar a pilha do usuário e do sistema;
- Mudar para o modo usuário;
- Desviar o fluxo para a função `user_code` do programa do usuário;

Descrição dos Periféricos

O *self-driving car* é equipado com diversos dispositivos para permitir sua locomoção segura, abaixo eles são brevemente descritos. Na seção seguinte são apresentados os detalhes para a interação com estes dispositivos (por meio de registradores mapeados em memória)

General Purpose Timer

O General Purpose Timer, ou GPT, é um periférico utilizado para obter o tempo atual do sistema. O tempo do sistema é monotonicamente crescente, desde quando o simulador foi iniciado.

Sensores de Luminosidade

O *self-driving car* possui sensores de luminosidade apontados para o chão, aproximadamente 30 centímetros à frente do carro, para que a luminosidade da pista seja lida, permitindo detectar as faixas da pista. Desta forma, a cada leitura deste sensor são 256 *bytes* retornados, onde cada *byte* varia de 0 a 255, sendo que valores mais próximos de 0 são mais escuros e próximos de 255 mais claros. Além disso, os índices mais baixos deste vetor representam as leituras mais à esquerda do carro e índices mais altos, as leituras à direita.

Sensor Ultrassônico

Para a detecção de objetos, o *self-driving car* possui um sensor ultrassônico que detecta objetos até 20 metros de distância à sua frente. Este sensor pode ser utilizado para evitar colisões com objetos no ambiente.

Global Positioning System e Giroscópio

O *Global Positioning System*, ou GPS, é um dispositivo anexado ao *self-driving car* permitindo com que a posição atual do carro em relação ao ambiente seja obtida. O *self-driving car* também possui um giroscópio, permitindo que a rotação do carro seja obtida.

Mapeamento dos Registradores dos Periféricos na Memória

Após carregar o dispositivo *Self driving car* e o *GPT* (mais detalhes na próxima Seção), os registradores dos dispositivos estarão mapeados em endereços de memória, mostrados na Tabela abaixo. “Valor” representa o valor lido ou escrito no/do endereço de memória. A coluna tamanho indica o tamanho da escrita/leitura que deve ser realizada em um dado endereço.

Endereço	**Tamanho**	**Função**
`0xFFFF0100`	`byte`	Quando atribuído valor 1, inicia a leitura da posição e rotação do carro pelo GPS. Quando o carro terminar de efetuar as leituras, o valor 0 é atribuído a este registrador.
`0xFFFF0101`	`byte`	Quando atribuído o valor 1, inicia a leitura do sensor de luminosidade. Quando a leitura dos sensores é finalizada, o valor 0 é atribuído a este registrador.
`0xFFFF0102`	`byte`	Quando atribuído o valor 1, inicia a leitura da distância do sensor ultrasônico. Quando o carro terminar de efetuar as leituras, o valor 0 é atribuído a este registrador.
`0xFFFF0104`	`palavra`	Após a leitura da posição e rotação ser realizada, i.e., quando o valor de `0xFFFF0100` for 0, este registrador armazena o valor do ângulo de Euler da rotação do carro no eixo x

0xFFFF0108	palavra	Após leitura da posição e rotação ser realizada, i.e., quando o valor de 0xFFFF0100 for 0, este registrador armazena o valor do ângulo de Euler da rotação do carro no eixo y
0xFFFF010C	palavra	Após leitura da posição e rotação ser realizada, i.e., quando o valor de 0xFFFF0100 for 0, este registrador armazena o valor do ângulo de Euler da rotação do carro no eixo z
0xFFFF0110	palavra	Após leitura da posição e rotação ser realizada, i.e., quando o valor de 0xFFFF0100 for 0, este registrador armazena o valor da posição do carro no eixo x, em decímetros
0xFFFF0114	palavra	Após leitura da posição e rotação ser realizada, i.e., quando o valor de 0xFFFF0100 for 0, este registrador armazena o valor da posição do carro no eixo y, em decímetros
0xFFFF0118	palavra	Após leitura da posição e rotação ser realizada, i.e., quando o valor de 0xFFFF0100 for 0, este registrador armazena o valor da posição do carro no eixo z, em decímetros
0xFFFF011C	palavra	Após realizada a leitura da distância do sensor ultrassônico, i.e., quando o valor de 0xFFFF0102 for 0, este registrador armazena a distância sensor de ultrassom, em centímetros.
0xFFFF0120	byte	Controla a direção horizontal do carro. Quando atribuído um valor negativo, o carro se deslocará para direita e quando atribuído um valor positivo, o carro se deslocará para a esquerda.
0xFFFF0121	byte	Controla a direção vertical do carro. Quando atribuído o valor 1, o carro andará para frente e quando atribuído o valor de -1, o carro andará para trás.
0xFFFF0122	byte	Controla o freio de mão do carro. Quando atribuído o valor 1, o freio de mão é acionado.
0xFFFF0124	256 bytes	Após finalizada a leitura do sensor de luminosidade, i.e. quando o valor de 0xFFFF0101 for 0, a partir deste endereço estará armazenado os valores da leitura do sensor de luminosidade. São 256 *bytes* onde cada *byte* varia de 0 a 255, sendo que valores mais próximos de 0 são mais escuros e próximos de 255 mais claros. Além disso, os índices mais baixos deste vetor representam as leituras mais à esquerda do carro e índices mais altos, as leituras à direita.
0xFFFF0300	byte	Quando atribuído o valor 1 inicia a leitura do tempo atual do sistema. Quando finalizada a leitura, o valor 0 é atribuído a este registrador
0xFFFF0304	palavra	Quando finalizada a leitura do tempo atual do sistema, i.e.

		quando o valor de `0xFFFF0300` for 0, este registrador armazena o tempo atual do sistema, em milissegundos. Este tempo é monotonicamente crescente, desde quando o simulador foi iniciado.

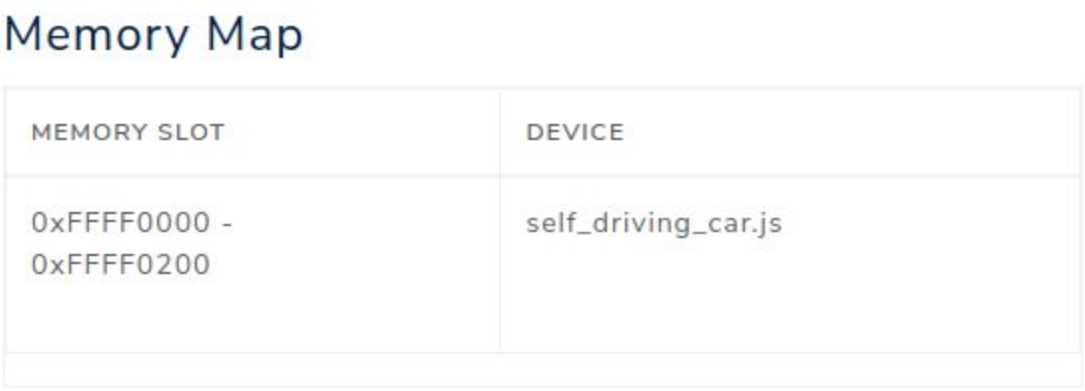
Realizando leitura de um registrador mapeado em memória

Para comunicação com os dispositivos periféricos utilizando registradores mapeados em memória, você deve utilizar a técnica de espera ocupada aguardando que o dispositivo responda. Por exemplo, para ler o valor da posição do carro no eixo x, o seguinte algoritmo pode ser utilizado:

1. Armazenar o valor 1 em `0xFFFF0000` (1 *byte*)
2. Ler o valor de `0xFFFF0000` (1 *byte*) até que o valor seja 0
3. Ler o valor de `0xFFFF0110` (4 *bytes*) que contém o valor da posição de x

Infraestrutura

Neste trabalho, você deve utilizar dois dispositivos externos: o *self-driving car* e o *General Purpose Timer (GPT)*, ligados ao processador RISC-V. O *self-driving car* que deverá estar mapeado ao Memory Slot de `0xFFFF0100` a `0xFFFF0300`, conforme a Figura abaixo. O GPT deve ser mapeado no Memory Slot de `0xFFFF0300` a `0xFFFF0500`. Ainda mais, no simulador, na aba Operating System**, a opção "*Enable Operating System*" deve ficar desabilitada**



No simulador, os valores obtidos pelo sensor de luminosidade do carro são mostrados por uma barra, no canto superior da tela do simulador. Note que, apesar da barra no canto superior direito da tela mostrar os valores obtidos pelo sensor de forma colorida, os valores que você irá trabalhar estão em escala de cinza, onde valores próximos de 0 são mais escuros e valores próximos de 255 mais claros.

Além disso, as linhas azuis à frente do carro mostram os valores obtidos pelo sensor ultrassônico. Quando as linhas ficam vermelhas, indica que algum objeto foi detectado à frente.

Entrega

Você deve submeter três arquivos: `loco.c, bico.s e soca.s` cada um deles adaptados com sua solução, conforme descrito nas seções acima. Assim, **você não deve submeter outros arquivos**. Além disso, no código auxiliar disponibilizado há um `Makefile` que deve ser utilizado para a compilação do seu programa e que será utilizado para avaliação. Você não deve alterar o conteúdo do `Makefile` ou dos outros arquivos que não devem ser submetidos (e.g., `api_car.h`).

Dicas e Observações

- Habilite a opção "*Enable debug controls*" para conhecer a pista. Para tanto, utilize as setas direcionais ou as teclas W, A, S e D. Note que esta opção deve ficar desabilitada durante o teste/execução de seu código.
- O ponto de início e destino são **fixos**.
- Tente utilizar uma aba anônima caso tenha problemas para carregar/utilizar o *self-driving car*
- O testador deste laboratório não detectará que o carro saiu da pista, apenas que chegou ao seu destino. Desta forma, atente-se para que o carro não saia da pista ou bata em algo.
- Sair da pista não quer dizer estar fora das linhas brancas, mas sim passar com o carro fora da calçada, isto é, na areia.
- A espera ocupada pode interferir no desempenho do simulador por realizar muitas requisições no barramento. Desta forma, no simulador, na aba *Hardware*, você pode alterar a frequência do barramento. Sugerimos a frequência de aproximadamente 10Hz para realizar testes e depuração. Para avaliação, utilizaremos o valor de 50Hz.
- Para fins de depuração, sugerimos que testem seus módulos separadamente. Por exemplo, implemente a *syscall* dos motores e faça um pequeno arquivo para testa-lá e faça o mesmo para as demais *syscalls* e outros trechos de código.
- Note que, para esta parte do trabalho você, possivelmente, não precisará utilizar todas as *syscalls*. Desta forma, você pode implementar apenas aquelas que lhe sejam úteis.