PARTICIONAMENTO SW/HW

Primeira alternativa

- ROM: *é a memória que armazena a implementação em software, usando a linguagem de máquina, das tarefas críticas e não críticas;*
- CPU: *é o processador que permite executar o software das tarefas;*
- *Este modelo tem baixo custo mas pode não atender os requisitos de tempo para as tarefas críticas.*

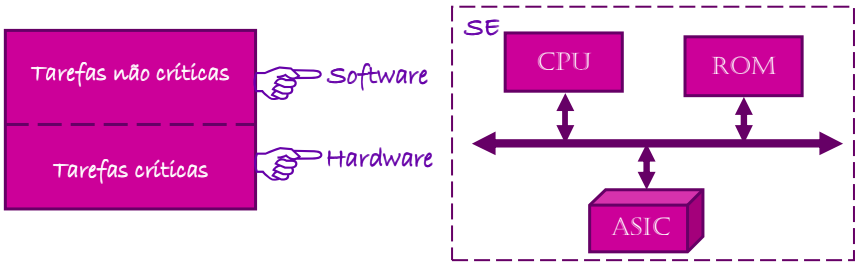
PARTICIONAMENTO SW/HW

Segunda alternativa

- HW: *é a implementação em hardware dedicada das tarefas críticas e das tarefas não críticas;*
- *Este modelo atende plenamente ao requisito de tempo para as tarefas críticas. No entanto, necessita muito esforço e consome muita área e por tanto tem muito alto custo.*

PARTICIONAMENTO SW/HW

Alternativa típica de um sistema embutido



- ASIC *ou* Application-Specific Integrated Circuit*: é a implementação em hardware das tarefas críticas;*
- ROM*: armazena a implementação em software, usando a linguagem de máquina, das tarefas não críticas;*
- CPU*: é o processador que permite executar o software e estabelecer a comunicação com o hardware.*

PARTICIONAMENTO SW/HW

- *O modelo típico de sistemas embutidos é chamado modelo integrado de hardware e software;*
- *O software escrito para sistemas embutidos, que não possuem armazenamento remoto, é geralmente chamado de firmware já que é embutido num chip de memória de somente leitura (ROM) ou Flash (EEPROM);*
- *O modelo da uma solução para atender o requisito de tempo imposto para as tarefas críticas;*
- *O esforço durante o projeto é razoável;*
- *Por tanto, o custo é também razoável.*

PARTICIONAMENTO SW/HW

- *Para avaliar a solução baseada num particionamento de SW/HW, precisa-se de um função objetivo que determina o quão boa essa solução.*
- *A solução pode ser avaliada termos de tempo de execução, área de hardware, consumo de energia e/ou dissipação de calor.*
- *Quando mais de uma característica deve ser levada em conta, pode-se também, dar mais ou menos importância à uma característica ou outra na formulação da função objetivo:*

  $F(P_{hw},P_{sw}) = w_1 \times$ área $+ w_2 \times$ tempo $+ w_3 \times$ energia $+ w_4 \times$ calor, onde w_i é um real positivo e $w_1 + w_2 + w_3 + w_4 = 1$

- *Entende-se que quanto maior o valor do coeficiente w_i, mais importância tem a característica correspondente.*
- *Por exemplo, com $w_1 = 0{,}75$ $w_2 = 0{,}25$ e $w_3 = w_4 = 0$, indica que precisa dar 3 vezes mais importância a área consumida que o tempo gasto e ignorar a energia consumida e o calor dissipado.*

PARTICIONAMENTO SW/HW

Algoritmos Gulosos de particionamento

```
Algoritmo Mudança Gulosa(P);
 P_HW := P;
 P_SW := ∅;
 Repita
   P_original = P_HW;
   Para cada objeto o_i ∈ P_HW Faça
     Se F(P_HW\{o_i},P_SW∪{o_i})< F(P_HW, P_SW)
     Então
         P_HW := P_HW\{o_i};
         P_SW := P_SW∪{o_i};
     Fim Se;
   Fim Para;
 Até P_HW = P_original;
```

PARTICIONAMENTO SW/HW

❶ *O algoritmo guloso começa com todos componentes do sistema na partição de hardware;*

❷ *Todo objeto que, se transportado para a partição de software permitiria melhorar alguma característica da solução (aquela que seria baseada nas partições atuais de hardware e software), é retirado da partição de hardware e inserido na de software;*

❸ *Se houver mudança no conteúdo da partição de hardware após a consideração de todos os objetos nela contidos, o tratamento da etapa ❷ é repetido, senão as partições atuais de software e hardware formam o resultado final de particionamento.*

- *Esse algoritmo é eficiente mas não considera as restrições impostas para o projeto.*

PARTICIONAMENTO SW/HW

```
Algoritmo Vulcan II(P);
 P_HW := P;
 P_SW := ∅;
 Repita
  P_original = P_HW;
  Para cada objeto o_i ∈ P_HW Faça
     TentaMudar(P_HW, P_SW, o_i)
  Fim Para;
 Até P_HW = P_original;

 Procedure TentaMudar(H, S, o)
  Se Restrição(H\{o_i},S∪{o_i}) e ℱ(H\{o_i},S∪{o_i})<ℱ(H, S)
  Então H := H\{o_i};
        S := S∪{o_i};
        Para cada objeto o_j ∈ sucessores(o_i) Faça
            TentaMudar(H, S, o_j);
        Fim Para;
  Fim Se;
```

PARTICIONAMENTO SW/HW

- *Função `Sucessores(O)` retorna todos os componentes que tem comunicação direta com O;*
- *Função `Restrição(H,S)` verifica se a partição de hardware `H` e a de software `S` atendem a restrição de desempenho imposta.*
- *O algoritmo cria uma partição de hardware que contem todos os componentes do sistema. Este particionamento garante que a restrição de desempenho é atendida;*
- *Todo componente do sistema que pode ser implementado em software, sem prejudicar o desempenho e que diminui o custo da implementação, é então retirado da partição de hardware e inserido na de software;*
- *Cada vez que um objeto O é removido da partição de hardware, o algoritmo tenta primeiro remover o objetos mais próximos de O.*

PARTICIONAMENTO SW/HW

Algoritmos de *Hill-Climbing*

- *Os algoritmos gulosos são eficiente mas não conseguem escapar de mínimo local porque rejeitam toda solução que não atende as restrições impostas;*
- *Para ultrapassar essa limitação, os algoritmos de particiona-mento precisam as vezes aceitar algumas mudanças apesar de serem negativas quanto ao desempenho da solução obtida.*
- *Ao contrário dos algoritmos gulosos, os algoritmos baseados em hill-climbing começam com todos os objetos na partição de software.*
- *O algoritmo extrai objetos da partição de software para a de hardware para conseguir atingir as metas ou a restrições impostas.*
- *Tal estratégia pode resultar em menos hardware que a estratégia usada pelos algoritmos gulosos.*

PARTICIONAMENTO SW/HW

Função Objetiva para algoritmos de *hill-climbing*

Suponha que as restrições impostas são para cada objeto O no projeto:

$$desempenho(O) \geq R_O$$

Então a função objetivo será definida como abaixo:

$$F_{obj}(H,S) = k_d \times \sum_{O \in S} violação(O) + k_t \times \sum_{O \in H} tamanho(O)$$

$$violação(O) = \begin{cases} 0 & desempenho(O) \geq R_0 \\ R_o - desempenho(O) & Senão \end{cases}$$

PARTICIONAMENTO SW/HW

```
Algoritmo Simulated-Anealing(P);
  P_HW := ∅; P_SW := P;
  temp := τ0; custo := Fobj(P_HW, P_SW);
  Enquanto (temp > τc) Faça
    Enquanto (não Equilíbrio) Faça
      (HWt,SWt):= MudaAleatória(P_HW,P_SW);
      Custot := Fobj(HWt, SWt);
      Δcusto := custot – custo;
      Se (Aceita(Δcusto,temp) > random(0,1)
      Então  (P_HW,P_SW):=(HWt,SWt);
               custo := custot;
      Fim se;
    Fim Enquanto;
    temp := α × temp;
  Fim Enquanto;
```

PARTICIONAMENTO SW/HW

- *O algoritmo* `Simulated-Anealing` *simula um processo físico aonde um material is derretido e seu estado de energia mínima é atingido baixando a temperatura o devagar suficiente para atingir um equilíbrio em cada temperatura.*
- *As constantes* τ_0 *e* τ_c *indicam as temperaturas inicial e de congelamento, previamente conhecidas;*
- *A função* `Equilíbrio` *determina se o processo de partitionamento para a temperatura corrente chegou a o equilíbrio, que pode ser aproximado pelo fato que no houve melhoria durante um certo número de iterações;*
- *A função* `Aceita(`Δ`,` τ`)` *determina se aceita a mudança ou não é calculada por* `min(1,` $e^{-\Delta/\tau}$`)`*, que retorna 1 quando o custo melhorou e então* $\Delta \leq 0$*;*
- *A constante* α *deve ter um valor em* `]0,1[`