**Sistemas Embarcados** (conceito e objetivos)

Definição:

Sistema embarcado é um sistema microprocessado, onde o conjunto é totalmente encapsulado e dedicado ao sistema que deve controlar. É um sistema completo e independente, desenvolvido para realizar tarefas bem específicas (e pré-definidas), cujo programa não pode ser acessado pelo usuário final, e que este poderá interagir através de interfaces, como displays, teclados, etc, desde que tenha sido projetado para tal finalidade.

Enquanto um computador usual (desktop/notebook/PDA) serve aos mais diversos propósitos: checar seu orkut/email, usar seu editor de texto, usar a Internet, etc., um sistema embarcado é implementado para um propósito específico.

Em geral, um sistema embarcado:

* É um sistema construído para executar uma tarefa, completamente ou parcialmente independente de intervenção humana.
* É especialmente desenvolvido para concretizar pequenas ações da forma mais eficiente possível.
* Interage com elementos físicos do ambiente. Ex: um sensor de temperatura.

Por definição, um sistema embarcado contém processador e software.

Certamente devido ao software também deve haver memória para armazenar o código executável e os dados temporários gerados durante seu funcionamento. Ela pode ser do tipo ROM ou RAM; mas geralmente todo sistema embarcado possui ambas. Se apenas uma pequena quantidade de memória for necessária, ela estará contida no mesmo chip do processador .Caso contrário, as duas se encontrarão em chips de memória externos.



Exemplo genérico de um sitema embarcado

Uma grande parte dos sistemas embarcados são sistemas de tempo crítico ou tempo real (time critical or real-time applications), isto é, são sistemas que funcionam em ambientes nos quais o controle de tempo (timing) é muito importante: os resultados de determinada operação só são relevantes se ocorrerem num determinado tempo. Um sistema de controle de vôo de um avião é um sistema embarcado de tempo crítico: em caso de algum problema durante o vôo, se o dispositivo de controle não detectar o problema prontamente e solucioná-lo de forma adequada, em milissegundos, haverá conseqüências catastróficas.

É interessante notar que um computador de propósito geral é composto de uma série de sistemas embarcados: teclado, mouse, HD, modem, sound card, etc. Cada um desses sistemas possui um processador e software embutidos, bem como tarefas específicas: o modem, por exemplo, envia e recebe dados de forma digital através da linha de telefone analógica. Se um sistema embarcado for bem projetado, a existência de um processador e de software pode passar completamente despercebida por um usuário de tal dispositivo. Esse é o caso do forno micro-ondas, do DVD, do relógio digital, da cafeteira, da máquina de lavar, do controle remoto, entre outros. Em alguns casos, seria possível construir um dispositivo equivalente, que não contivesse um processador nem software. Isso poderia ser feito substituindo tal combinação por um circuito integrado que realizasse as mesmas tarefas em hardware puro. Entretanto, muita flexibilidade se perde, pois é muito mais fácil e mais barato mudar algumas linhas de código, do que projetar novamente uma peça de hardware. Além disso, de maneira geral os sistemas embarcados não possuem muita flexibilidade (nem de software, nem de hardware).

A única flexibilidade permitida é o caso de upgrades para novas versões, fazendo com que o sistema possa ser reprogramado, com correções de bugs e implementação de novas funções emandadas pelo mercado (através do chip FPGA, por exemplo). Essas atualizações são, na maioria as vezes, feitas pelos fabricantes, e em casos particulares pelos usuários finais.

**HISTÓRIA DOS SISTEMAS EMBARCADOS**

Nos primeiros anos dos computadores digitais na década de 1940, os computadores eram por vezes dedicados a uma única tarefa. Eram, entretanto, muito grandes e caros, comparados com os sistemas embarcados de hoje em dia.

O termo sistema embarcado tem sua origem no fim da década de 1960. Nessa época o que

existia era um pequeno programa de controle funcional de telefones, escrito em assembler, que era usado em outros dispositivos (não eram considerados um sistema embarcado em si).

Também na década de 60 surgiu o primeiro sistema embarcado reconhecido, durante o desenvolvimento do Projeto Apollo (conjunto de missões espaciais da Nasa para enviar o homem à Lua). Esse sistema era o Apollo Guidance Computer (usado para coletar e fornecer informações sobre voos, alem de controlar automaticamente todas as funções de navegação) , desenvolvido por Charles Stark Draper (1901 - 1987) no MIT.

O primeiro sistema embarcado de produção em massa foi o computador guia do míssil nuclear LGM-30 Míssil Minuteman, lançado em 1961 pelos EUA. Ele possuía um disco rígido para a memória principal e era usado para controlar a orientação e estabilidade do foguete. Quando a segunda versão do míssil entrou em produção em 1966, o computador guia foi substituído por um novo, que constituiu o primeiro uso em grande volume de circuitos integrados (reduzindo o preço dos mesmos). A construção desse sistema imbarcado foi muito importante, pois ele estimulou as pesquisas com relação aos custos dos sistemas, diminuindo o preço dos mesmos.

Outro sistema embarcado utilizado em voos foi o sistema embarcado do caça Tomcat F-14 da marinha americana. Esse caça, lançado em 1970, continha o primeiro microprocessador baseado em sistema embarcado. O sistema era chamado de Garret CADC, controlava boa parte das funções do caça e tinha 8 processadores e 19 chips de memória.

Na década de 1970 começavam a surgir bibliotecas de códigos direcionados para

sistemas embarcados específicos com processadores específicos. Atualmente os sistemas

embarcados podem ser programados em linguagens de alto nível e possuem sistemas operacionais, facilitando a sua programação.

Durante a década de 1970, os sistemas embarcados dó evoluíram graças a indústria de defesa militar. Todavia, devido a difusão dos sistemas embarcados, outras áreas estimulam o desenvolvimento desses sistemas. São elas: a indústria de jogos, a medicina e a aviação.

Desde suas primeiras aplicações, os sistemas embarcados vêm reduzindo seu preço e aumentando o seu poder de processamento e funcionalidade, principalmente após a década de 80. Nessa década, vários componentes externos foram integrados no mesmo chip do processador, o que resultou em circuitos integrados chamados microcontroladores. Esses microcontroladores não precisvam de ferramentas periféricas para executar o seu trabalho. Microcontroladores são mais baratos, porém menos flexíveis do que os tradicionais microprocessadores, mas mesmo assim essa ferramenta tecnológica possibilitou a difusão dos sistemas embarcados, estes atuando em areas que antes não se imaginaria um computador trabalhando.

Antigamente, microcontroladores de 16 e 32 bits tinham muitas desvantagens. O seu desenvolvimento era caro, os dispositivos eram grandes e pouco potentes. Por isso, era comum se utilizar microcontroladores de 8 bits apenas. Todavia, hoje em dia os preços caíram substancialmente e os dispositvos evoluíram. Assim, os microcontroladores de 32 bits passaram a ser utilizados, o que difundiu ainda mais os sistemas embarcados, pois estes puderam entrar em áreas que antes não eram aplicados, pois eram muito lentos.

Com o custo de microcontroladores menor que um dólar americano, tornou-se viável substituir componentes analógicos caros como potenciômetros e capacitores por eletrônica digital controlada por pequenos microcontroladores. No final da década de 1980, os sistemas embarcados já eram a norma ao invés da exceção em dispositivos eletrônicos.

Outro fator que ajudou na difusão dos sistemas embarcados foi a criação do Consórcio PC/104 pela Ampro, RTD e outros fabricantes. Esse grupo estabeleceu estabeleceu um formato para microprocessadores Intel baseado em uma placa-mãe de aproximadamente quatro polegadas quadradas, e um pouco menos de uma polegada de altura. As tábuas foram empilhadas, permitindo que um computador muito potente pudesse ser montado em uma caixa de aproximadamente quatro polegadas quadradas. O PC/104 foi inicialmente dirigido aos militares e às áreas relacionadas à saúde, onde tornou-se amplamente utilizado. Quando o poder do processador cresceu suficientemente lidar com aplicações multimídia, o PC/104 passou a ser utilizado em outras áreas.

[](http://pt.wikipedia.org/wiki/Ficheiro:Apollo_program_insignia.jpg) Insignia do projeto Apollo, que desenvolveu o primeiros sistema embarcado

[](http://en.wikipedia.org/wiki/File:Charles_Stark_Drape.jpg)Charles Stark Draper, criador do primeiro sistema embarcado.

[](http://en.wikipedia.org/wiki/File:Dsky.jpg) interface do Apollo Guidance Computer

O caça F-14, controlado por um sistema embarcado.

**Programando SE:**

**INTRODUÇÃO:**

Programar sistemas embarcados não é como programar um PC comum. De fato programação para um sistema embarcado se assemelha a programação de computadores de 15 anos atrás. O hardware escolhido para o sistema é em geral escolhido para que o dispositivo seja o mais barato possível. Bem, isso significa que programador terá que lidar com processadores lentos e baixa memória, enquanto que ao mesmo tempo se esforçará para conseguir uma eficiência que não é vista na maioria dos aplicativos para PC.

Há um grande número de arquiteturas diferentes de sistemas embarcados e isso torna as ferramentas para programar mais caras. Além disso, essas ferramentas têm menos recursos e são menos desenvolvidas. Num projeto embarcado de grande porte, fatalmente você irá encontrar algum ‘’bug’’no compilador. Ferramentas de “debug” são outro problema, pois nem sempre você consegue rodar um “debugger” no seu processador embarcado devido à escassez de recursos e isso torna o controle de erros do programa difícil. Hardwares especiais como portas JTAG podem superar esse problema parcialmente. Debugging é de suma importância, pois se o sistema “travar” durante o funcionamento de um sistema de hardware real (como um motor, por exemplo), podem ocorrer danos permanentes ao equipamento.

Para um programador de sistemas embarcados, fazer uso dos algoritmos mais eficientes é essencial, bem como um entendimento da arquitetura de hardware sobre a qual você está trabalhando, de forma a facilitar a otimização do seu software.

Memória também é um problema, por ser escassa: diferentemente de programas para PC, o programador deve usar algoritmos que sacrifiquem tempo do processador em favor da memória. Ou seja, nada de “memory leak” (consumo de memória, não intencional, quando um programa falha em liberar memória que não é mais necessária). Frequentemente, aplicativos embarcados usam técnicas determinísticas de memória e evitam o uso de funções “malloc” (função C de alocação dinâmica), facilitando a investigação e solução de possíveis “leaks”.

**C e Assembly**

Muitos programadores preferem programar em C a programar em assembly, por boas razões, já que C é de alto nível e portanto libera o profissional do conhecimento e controle de alguns detalhes de implementação a nível de máquina.

Contudo há alguns processos de baixo-nível que ou podem ser melhor implementados em assembly ou **somente** podem ser implementados em linguagem assembly. Por outro lado é freqüentemente útil para o programador analisar o código assembly produzido pelo compilador C e manualmente editá-lo a fim de otimizá-lo de formas que o compilador não consegue. Assembly também é útil para processos de tempo crítico (time-critical) porque diferentemente do que ocorre com linguagens alto-nível não há ambigüidade quanto a como o código será compilado. O “timing” pode ser controlado de forma mais eficiente, o que é útil para produzir “drivers” simples.

A maioria dos programadores de C são “mal-acostumados” porque eles programam em ambientes nos quais há uma série de bibliotecas prontas para uso. O fato cruel é que em sistemas embarcados raramente há tantas bibliotecas às quais o programador habituou-se a usar, ocasionalmente um sistema embarcado pode nem ter uma biblioteca padrão completa, se sequer houver uma. Frequentemente, devido a problemas de espaço, não é possível incluir uma biblioteca inteira, e programadores são muitas vezes forçados a implementar suas próprias bibliotecas. Enquanto algumas bibliotecas são muito grandes e não muito adequadas ao uso em microcontraladores, muitos sistemas de desenvolvimento incluem as bibliotecas padrão mais comuns para programadores C.

C continua a ser uma linguagem muito popular para microcontroladores devido à sua eficiência de código e reduzido tempo de desenvolvimento. C oferece controle de baixo nível e é mais humanamente legível que assembly. Muitos compiladores C estão disponíveis para uma grande gama de plataformas de desenvolvimento. Os compiladores são parte de IDEs com uma janela de assembly. Adicionalmente, usando C há um aumento na portabilidade, uma vez que o código em C pode ser compilado por diferentes tipos de processadores.

## Bootloader

A fim de simplificar muitas tarefas, programadores de muitos sistemas usam um tipo genérico de software chamado **bootloader** que é responsável por realizar algumas rotinas do sistema (como: liberar o modo protegido), e então carregar o kernel na memória, para a partir daí transferir o controle do sistema para o kernel. Bootloaders são usados em muitos microcontroladores e são em geral a forma mais rápida para atualizar um programa, com pequenas mudanças, em um microcontrolador. Isso torna o ciclo editar-compilar-baixar-testar um pouco mais rápido.

**Ferramentas de Desenvolvimento:**

**O proceso de construção**

Existem muitas coisas que as ferramentas de desenvolvimento de software podem fazer automaticamente quando a plataforma alvo está bem definida. Essa automação é possível porque as ferramentas podem explorar recursos do hardware e do sistema operacional nos quais o programa será executado. Dessa forma o compilador, esconde do programador certos aspectos do processo de construção do software.

O processo de conversão do código-fonte para do software embarcado em um arquivo binário executável envolve três passos distintos. Primeiro, cada arquivo-fonte precisa ser compilado ou montado em um programa-objeto. Depois, todos os os programas-objetos originados a partir do primeiro passo necessitam ser interconectados para produzir um objeto único, chamado programa-relocável (“ relocatable program”). Por último, o endereço físico da memória devem receber os “offsets relatives” do programa-relocável no processo chamado realocação. O resultado do terceiro passo é um arquivo que contém um um programa-executável em binário que está pronto para ser rodado no sistema embarcado.



Figura: Processo de conversão do código-fonte para um arquivo binário do Sistema Embarcado

Cada um dos passos no processo de construção do software embarcado é uma transfromação realizada por software rodando num computador de propósito geral. Para distinguir esse computador, onde se realiza o desenvolvimento ( PC ou Unix workstation), do sistema embarcado alvo (target), ele é referido como o computador host. Em outras palavras, o compilador, assembler, linker são softwares rodando no host, e não no sistema alvo. Ainda assim, apesar do fato de rodarem em outra plataforma computacional, essas ferramentas se combinam para produzir um executável binário que irá rodar apropriadamente no sistema embarcado alvo.

Segundo uma pesquisa realizada recentemente pela www.8052.com, 49% dos usuários de

processadores compatíveis com o 8051 usam assembly, 33% usam C, 5% usam Basic, 3% usam

Pascal e 9% usam outras linguagens. O interesse por linguagens orientadas a objeto vem crescendo

dia a dia. Tal crescimento de popularidade é devido às melhorias que o paradigma de orientação a objetos proporciona

ao ciclo de desenvolvimento, tornando- o mais rápido como o uso de melhores métodos de estruturação e modularização,

bem como a reutilização de objetos. No entanto, há ainda certos obstáculos a serem superados para aumentar a efetividade

de tal paradigma no âmbito dos sistemas embarcados. Linguagens orientadas a objetos criam e destróem objetos dinamicamenteme

dificultando o controle de memória, bem como do tempo de execução dos programas. Linguagens como Java que usam Garbage

Collectors automáticos podem tornar inviável o controle do tempo de execução (que é crítico em sistemas de tempo real),

uma vez que não sabemos a princípio quando o garbage collector será acionado e quanto tempo durará sua execução.

Estes problemas estão sendo abordados pelos desenvolvedores da linguagem e por pesquisadores de um modo geral.

****

**Divisão entre host e alvo**

Vale ressaltar o fato de os kernel’s monolíticos vem se tornando cada vez mais populares por seus vários benefícios. É um kernel relativamente grande com capacidades sofisticadas é adaptado para um ambiente embarcado. Isso dá ao programador um ambiente similar ao de um sistema operacional desktop como o [Linux](http://en.wikipedia.org/wiki/Linux) ou o [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows), e, portanto o desenvolvimento é muito produtivo, por outro lado, tal kernel requer consideravelmente mais recursos de hardware, e por causa da sua complexidade esse tipo de kernel é menos previsível e menos seguro/confiável (maior chance de erros). Alguns exemplos de kernel’s monolíticos embarcados são [Embedded Linux](http://en.wikipedia.org/wiki/Embedded_Linux) e o [Windows CE](http://en.wikipedia.org/wiki/Windows_CE). Apesar de aumentar o custo do hardware, esse tipo de sistema embarcado está cada vez mais popular, especialmente em dispositivos mais poderosos como [Wireless Routers](http://en.wikipedia.org/wiki/Routers) e [GPS Navigation Systems](http://en.wikipedia.org/wiki/Automotive_navigation_system).

Há diferentes tipos de arquitetura de software em uso:

**SUPER LOOP:**

Um superloop é uma estrutura de programa composto de um loop infinito, com todas as tarefas do sistema contidas nesse loop. Aqui está um pseudocódigo de um superloop.

Function Main\_Function()

{

Initialization();

Do\_Forever

{

Check\_Status();

Do\_Calculations();

Output\_Response();

}

}

Nós colocamos as rotinas de inicialização antes do superloop, porque nós somente queremos inicializar o sistema uma única vez. Assim que o loop infinito seja iniciado, nós não queremos “resetar” os valores, porque precisamos manter estado persistente no sistema embarcado. Dessa forma, torna-se imprescindível o uso do superloop nos sistemas embarcados. Sistemas embarcados não são os únicos a utilizar tal arquitetura, jogos, por exemplo, constantemente usam um loop similar, o chamado *(tight) (main) game loop*.

## Power Save Superloop:

Digamos que tenhamos um sistema embarcado que tem um tempo de loop de 1ms, e só precisa checar certa entrada de dados uma vez por segundo. É um evidente gasto continuar a repetir o loop ininterruptamente, especialmente se não precisamos fazer nada na maior parte do tempo. Nessa situação, por exemplo, o programa irá executar 1000 loops antes de ler a entrada, e os outros 999 loops terão sido apenas uma contagem regressiva para a próxima leitura. Nesse caso, é muito ineficiente ter o processador rodando a 100% todo o tempo. Seria muito melhor implementa um atraso (delay) no programa, tornando esse sistema embarcado mais eficiente. O superloop extendido seria:

Function Main\_Function()

{

Initialization();

Do\_Forever

{

Check\_Status();

Do\_Calculations();

Output\_Response();

**Delay\_For\_Next\_Loop();**

}

}

Se tal atraso for de 999ms, não precisamos de 1000 loops e podemos ler a entrada em cada loop.

Desenvolvedores de sistemas embarcados usam [compiladores](http://en.wikipedia.org/wiki/Compiler), [assemblers](http://en.wikipedia.org/wiki/Assembly_language#Assembler) e [debuggers](http://en.wikipedia.org/wiki/Debugger) para desenvolver softwares de sistemas embarcados. Contudo, eles também usam ferramentas mais específicas. Entre elas estão ferramentas , matematicas como o MATLAB (software voltado para o calculo numerico), ou ainda se pode personalizar uma linguagem de programação para otimizar o seu uso (Java, Pascal ).

DEBUGGING

Como um sistema embarcado é geralmente composto por uma série de elementos, a estratégia de debugging pode variar. Por exemplo, debugar um sistema embarcado centrado em software-microprocessador é diferente de debugar um sistema no qual a maior parte do processamento é realizada por periféricos.

Sistemas embarcados freqüentemente residem em máquinas que têm expectativas de funcionamento contínuo por anos, sem erros, ou em alguns casos que se recuperem automaticamente de algum erro que ocorra. Portanto o software é geralmente desenvolvido e testado mais cuidadosamente do que se este fosse destinado a computadores pessoais.

Os softwares embarcados são muitas vezes construídos de forma que um erro possa ser recuperado automaticamente, pois o sistema está inacessível (exemplo: sistemas espaciais), ou precisa estar sempre ligado (sistema de controle de reatores nucleares), ou ainda o mau funcionamento da maquina pode acarretar grandes prejuízos (ex: caixas automáticos).

FPGA

FPGAs são chips digitais programáveis, ou seja, eles podem ser programados para realizar qualquer função digital, pois são compostos de um enorme número de chaves programáveis, que podem ser configuradas para simular o comportamento de qualquer outro circuito.

Internamente os FPGAs são compostos de blocos básicos, que por sua vez são compostos de uma lógica combinacional que pode serprogramada para implementar qualquer função booleana de 4 ou 5 variáveis, conforme o modelo usado. Cada bloco básico dispõe também de elementos de memória (flip-flops) que podemarmazenar os resultados obtidos pela função booleana. A conexão entre blocos básicos também é configurável, formando complexas estruturas combinacionais com armazenamento de estado nos blocos configurados como memória. Em resumo, é possível configurar o FPGA para executar qualquer tipo de algoritmo.

Após criar o programa desejado e convertê-lo em um arquivo binário com um compilador, podemos transferi-lo para o chip FPGA. Vale ressaltar que esse processo pode ser repetido várias vezes, logo o chip pode desempenhar diferentes funções dependendo da conveniência do usuário. FPGAs “perde” o programa quando é o fornecimento de energia é interrompido (como a memória RAM). Desta forma, para recuperar a antiga função, é necessário que o chip FPGA seja reprogramado.

Um único FPGA pode simular não apenas um processador simples, mas também outros circuitos de apoio, como o controlador de vídeo, uma interface serial e assim por diante. Existem vários tipos de FPGA, alguns com programação com tecnologia PROM, que só podem ser programados uma única vez, outros com EPROM, que podem ser reprogramados em laboratório, e ainda outros em RAM, onde é possível até reconfigurar durante a execução. Isso dá uma grande flexibilidade ao projetista aliado a uma performance em geral bem maior que a obtida com o uso de processadores de propósito geral.



*Chip FPGA*

* APLICAÇÃO

O crescente avanço das *Field Programmable*

*Gate Arrays* (FPGAs) tornou possível a criação

e utilização de *soft-cores*, processadores projetados

e implementados em linguagem de descrição de

hardware. A flexibilidade das FPGAs permite que o projetista

do processador possa adaptá-lo aos requisitos da

aplicação.

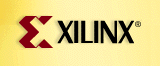
**SISTEMAS EMBARCADOS: ACOPLAMENTO DO SOFT-CORE PLASMA AO**

**BARRAMENTO OPB DE UM POWERPC 405**

*Rafael Vargas1, Rafael Cancian2, Hugo Marcondes3, Antônio A. Fröhlich4.*

Universidade Federal de Santa Catarina

Florianópolis, SC, Brasil

* Maiores produtores mundiais: Existem (no mínimo) 5 companhias produzindo FPGAs no mundo. As duas maiores são a (XILINX) [](http://www.xilinx.com/) e [http://www.fpga4fun.com/images/Altera_logo.gif](http://www.altera.com/) (ALTERA)

**Tendências Futuras**

Com o advento dos computadores e da tecnologia da informação em geral (Internet, notebooks, smartphones, etc), os sistemas embarcados estão ganhando força e tornando-se imprescindíveis para o desenvolvimento de novas tecnologias. Assim, eles tem deixado de ser criados apenas como dispositivos isolados, com baixa importância, e têm ganho uma maior abrangência de aplicações. Para tanto, é fundamental incentivar pesquisas acadêmicas e garantir incentivos para investimentos empresariais em prol de assegurar um acúmulo de conhecimentos sobre sistemas embarcados, que possibilite, no futuro próximo, avançar ainda mais nessa nova tecnologia.

Um sonho antigo que está se aproximando cada vez mais com a realidade é a possibilidade de comunicação entre os diversos aparelhos de fabricantes diferentes. Por exemplo, seria prático você poder controlar sua televisão através do celular ou que todos eletrodomésticos de sua casa possam ser manipulados do seu computador, onde quer que você esteja. Já existem inúmeras técnicas desenvolvidas com esta finalidade, como Bluetooth, porém ainda há muito trabalha a ser feito para que haja uma sincronia na atividade em conjunto dos equipamentos. Os sistemas embarcados, em particular, estão sendo desenvolvidos para se adaptar melhor às situação corriqueiras dos usuários, buscando aumentar nossa qualidade de vida na medida que pode ser responsável por facilidades no nosso dia-a-dia. Por isso os sistemas embarcados são tão importantes, sendo decisivos em relação a definição do futuro cenário tecnológico que precensiaremos dentre alguns anos. Enfim, há uma série de fatores que precisam ser melhorados e corrigidos para que os softwares embarcados possam desempenhar o seu papel de forma eficiente, entre os quais se destacam: tamanho, velocidade de processamente, falhas, flexibilidade, custo e energia.

**grupos de pesquisa e projetos brasileiros**

**Grupo de Sistemas Embarcados - GSE**

**Instituição: Pontifícia Universidade Católica do Rio Grande do Sul - PUC RS**

O Grupo de Sistemas Embarcados (GSE) realiza pesquisa e desenvolvimento na área de sistemas embarcados em geral, tanto a nível de software quanto de hardware. As áreas de atuação do grupo incluem aplicações espaciais (sistema de computação de bordo de satélites), mobilidade e pervasive computing, redes wireless ad-hoc (redes de sensores sem fio), tecnologia RFID, computação reconfigurável (VHDL/FPGA), e sistemas embarcados para agricultura de precisão. Entre as atividades de pesquisa em andamento destacam-se os trabalhos em cooperação com: Grupo de Sistemas, Sinais e Computação (SISC) da Faculdade de Engenharia da PUCRS; Grupo de Apoio ao Projeto de Hardware (GAPH), da Faculdade de Informática da PUCRS; Laboratório de Microgravidade do IPCT da PUCRS; Centro de Excelencia em Tecnologia Eletronica Avancada (CEITEC), Space Science Centre da University of Sussex, Inglaterra; Agência Espacial Brasileira (AEB) e Instituto Nacional de Pesquisas Espaciais (INPE). O grupo possui projetos financiados por órgãos de fomento como Finep, Fapergs, CNPq e Agência Espacial Brasileira, incluindo não apenas equipamentos e material permanente, mas também bolsistas de graduação e de pós-graduação. Os resultados das pesquisas do grupo resultam no desenvolvimento de produtos e serviços para aplicação direta pela sociedade. Os resultados mais relevantes das pesquisas são eventualmente aproveitados em produtos comerciais da empresa Innalogics, spin-off do GSE.

**Automação do Projeto de Sistemas Computacionais Embarcados**

Universidade Federal de Santa Catarina - UFSC

O grupo aborda temas das áreas de Microeletrônica, projeto auxiliado por computador (CAD/EDA), bem como software e hardware para sistemas embarcados. O impacto do trabalho deste grupo verifica-se em nove vertentes principais: 1-A modelagem de plataformas de sistemas integrados de hardware e software (SoCs) em vários níveis e estilos de descrição (TLM, funcional/comportamental e RTL). 2-A investigação de novas técnicas de otimização, análise, verificação e síntese a serem incorporadas em ferramentas de Electronic Design Automation (EDA); 3-A investigação de novas técnicas para acelerar o desenvolvimento, a depuração e a otimização de software dependente de hardware, tais como a o redirecionamento automático de montadores, ligadores, depuradores, escalonadores para diferentes processadores-alvo. 4-O desenvolvimento de software dependente de hardware (drivers, componentes de sistema operacional, firmware). 5-A concepção, a verificação e o teste de circuitos integrados digitais (ASICs) e de componentes (IPs) de sistemas integrados de hardware e software (SoCs). 6-O desenvolvimento de técnicas de projeto de circuitos digitais tolerantes a falhas temporárias. 7-O desenvolvimento de arquiteturas para aplicações específicas, em especial a compressão de imagens estáticas e dinâmicas. 8-O desenvolvimento de IPs e drivers voltados para aplicações em dispositivos móveis (em especial Linux-based phones). 9-A prototipação rápida de circuitos e sistemas embarcados através de FPGAs.

**Desenvolvimento e Otimização de Programas para Sistemas Embarcados baseado em Componentes de Software**

Universidade Federal do Amazonas - UFAM

O uso dos chamados componentes de software é um novo paradigma para o desenvolvimento de programas de computador que precisa ser mais bem elaborado de maneira a tornar-se um padrão de desenvolvimento de fato. Nesta abordagem constroem-se componentes de software reutilizáveis, que são amplamente testados, para depois colocá-los à disposição de outros desenvolvedores. De fato, os componentes de software assim utilizados tendem a possuir características de produto, com qualidade assegurada e funcionalidade comprovada. Estatísticas mostram, ainda, o crescente número de sistemas microprocessados que são colocados no mercado mundial a cada ano. De fato, a construção de microprocessadores e sistemas digitais é de fundamental importância para qualquer país que queira uma maior inserção no mercado mundial. Ainda observando o mercado nota-se que apenas aproximadamente dez por cento dos chips construídos são utilizados em computadores de mesa ou laptops. A grande maioria dos chips construídos é utilizada em sistemas menores, sistemas estes com pouco poder de processamento e limitada capacidade de memória. Tais dispositivos são nomeados na literatura de sistemas embarcados ou sistemas embutidos. O tema de interesse deste grupo de pesquisa aborda estas duas áreas do conhecimento tão importantes para a indústria mundial, Componentes de Software e Sistemas Embarcados. Nosso desafio é investigar e propor arquiteturas de componentes de software que tenha aplicações em sistemas embarcados de uso industrial. Para tanto, investigamos tecnologias de componentes de software comerciais que tem uso em sistemas embarcados já relatadas na literatura, as desenvolvidas especificamente para a indústria de sistemas eletrônicos de entretenimento, e as desenvolvidas em institutos universitários de pesquisa. Procuramos validar estas arquiteturas através da da implementação de aplicações reais para sistemas embarcados disponíveis no mercado.

**GPSE - Grupo de Pesquisa em Sistemas Embarcados**

Universidade de Pernambuco - UPE

Os resultados das pesquisas deste grupo abrangem contribuições teóricas e práticas com foco específico em Aplicações Embarcadas. Entre elas constam: Exploração de Arquiteturas e Mecanismos para redução de consumo de energia em aplicações embarcadas.

**Sistemas Embarcados**

Universidade Federal do Rio Grande do Sul - UFRGS

Tendo em vista a crescente dependência da sociedade no uso de recursos computacionais embarcados nos mais diversos sistemas e equipamentos (automóveis, telefones celulares, lavadoras de roupa, instrumentos biomédicos), há uma necessidade crescente de formação de pessoal na linha de pesquisa de projeto e integração completa de sistemas complexos. O grupo busca modelar e realizar automaticamente o projeto de sistemas computacionais embarcados, constituídos de combinações de componentes de hardware e software. Em especial, o trabalho é bastante focado em tecnologias e aplicações que tenham forte possibilidade de apropriação por empresas nacionais, combinado com a produção de trabalhos científicos com impacto internacional. Os tópicos de pesquisa cobrem a arquitetura de processadores dedicados e de sistemas multi-processadores, a geração automática do software embarcado, os sistemas operacionais embarcados de tempo real, as ferramentas de modelagem e validação de sistemas e o teste do software e do hardware dos sistemas embarcados.

**Perfil Curricular e Cadeiras Relevantes no Cin**

**Sistemas Embarcados - Cadeiras:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Códi-  go | Área especializante | CH semanal(Teo./Prat) | Crédi-  tos | CH total | Pré-requisitos | Co-requisitos |
| IF725 | Projeto de CPU | 3/2 | 4 | 75 | IF674 | IF827 |
| IF726 | Organização e Tecnologia de Computadores | 3/2 | 4 | 75 | IF674 |  |
| IF727 | Processadores de Aplicação Específica | 3/2 | 4 | 75 | IF827 IF725 IF726 IF709 |  |
| IF728 | Engenharia de Sistemas Embutidos | 3/2 | 4 | 75 | IF674 |  |
| IF709 | Implementação de Sistemas Operacionais | 3/2 | 4 | 75 | IF677 IF674 |  |
| IF729 | Prototipação de Circuitos Integrados | 3/2 | 4 | 75 | IF674 |  |
| IF730 | Sistemas de Tempo Real | 5/0 | 5 | 75 | IF677 |  |
| IF814 | Met. de Hardware/Software Co-design | 3/2 | 4 | 75 | IF674 |  |
| IF732 | Projeto de Sistemas Embutidos | 3/2 | 4 | 75 | IF729  IF728 |  |
| IF733 | Transdutores | 3/2 | 4 | 75 | ES238 |  |
| IF828 | Validação de Sistemas Embutidos | 3/2 | 4 | 75 | IF674 |  |
| IF829 | Tolerância a Falhas | 3/2 | 4 | 75 | IF674  IF677 |  |
| IF830 | Sistemas Assíncronos | 3/2 | 4 | 75 |  |  |
| IF831 | Arquiteturas Avançadas de Computadores | 5/0 | 5 | 75 | IF674 |  |
| IF832 | Medidas de radiofreqüência | 3/2 | 4 | 75 | FI108 |  |
| IF833 | Instrumentação Eletrônica | 3/2 | 4 | 75 | ES238 |  |
| IF735 | Tópicos Avançados em Arquitetura de Computadores | 5/0 | 5 | 75 | IF674 |  |
| IF737 | Tópicos Avançados em Sistemas Embutidos | 5/0 | 5 | 75 | IF674 |  |
| IF736 | Seminário em Arquitetura e Sistemas Embutidos | 3/0 | 3 | 45 | IF674 |  |