Introdução

O projeto do processador Nano tem por objetivo servir de exercício de fixação do conteúdo apresentado na disciplina de Circuitos Digitais, notadamente ao uso das linguagens de descrição de hardware, e servir também como uma introdução à arquitetura dos processadores.

O processador Nano é extremamente simples, tendo sua arquitetura sido desenvolvida tendo como base a arquitetura do processador MIPS. Por simplicidade, este opera com apenas 16 instruções de 16 bits e possui um banco de registradores com apenas 8 registradores de 8 bits.

O processador Nano pode ser classificado como sendo da categoria dos processadores RISC e, a semelhança do processador MIPS, pode ter suas instruções organizadas em três grupos distintos denominados de instruções tipo R, tipo I e tipo J.

As instruções tipo R são aquelas em que os operandos se encontram carregados no banco de registradores. As instruções tipo I são aquelas em que um dos operandos é passado diretamente na palavra da instrução. As instruções tipo J são aquelas utilizadas para representar as instruções de desvio incondicional. A Figura 1 a seguir apresenta o formato interno de cada uma destas instruções.

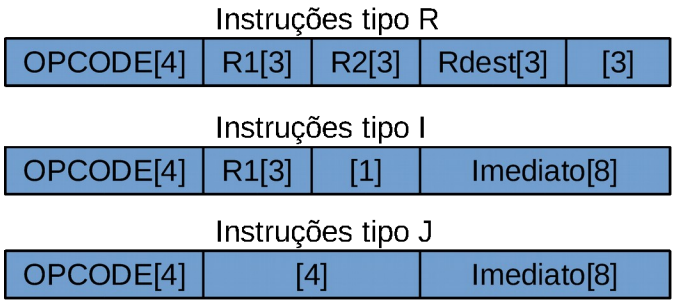


Figura 1: Formato das instruções do processador

Como se pode observar, todas as instruções possuem um campo denominado de OPCODE, o qual é utilizado para identificar cada uma das 16 instruções.

Os campos R1 e R2 são utilizados para identificar a localização onde o primeiro e o segundo operando das instruções tipo R estão armazenados no banco de registradores. No caso das instruções tipo I, o primeiro operando é apontado pelo campo R1 e o segundo operando é carregado diretamente no campo imediato. Por fim, as instruções tipo J possuem apenas um operando, o qual é carregado diretamente no campo imediato, e serve para indicar em quantas instruções a execução do programa deve ser desviado.

Os valores carregados no campo imediato, tanto para instruções tipo I quanto para instruções tipo J, é representado em complemento a dois, ou seja, podem representar tanto valores positivos quanto negativos.

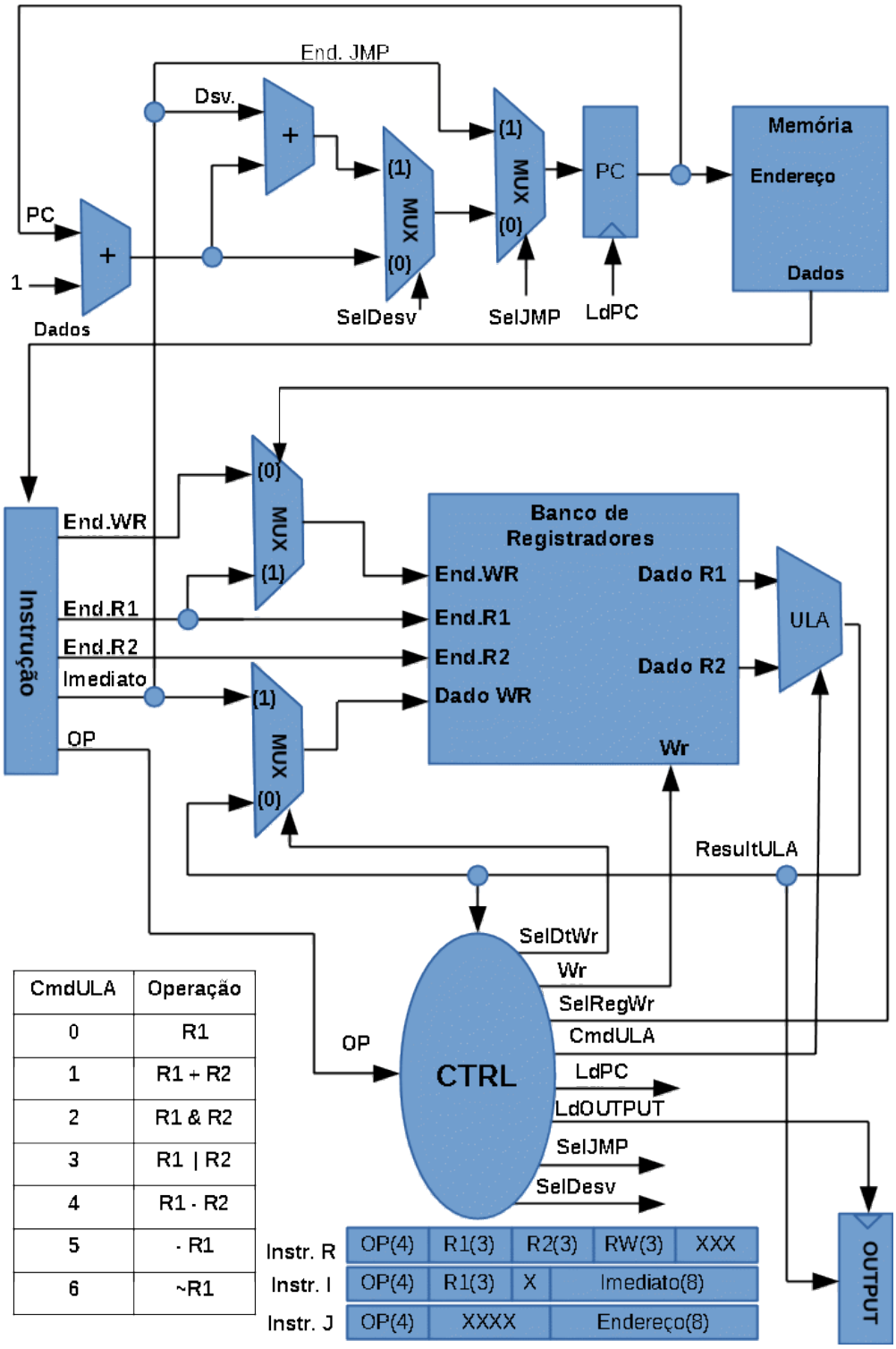
A tabela a seguir traz uma listagem completa com todas as 16 instruções, seu formato e seu modo de operação.

OPCODE	Instrução	Fmt.	Operação
0000	NOP	R	Não executa nenhuma operação
0001	ADD	R	R[Rdest] = R[R1]+ R[R2]
0010	AND	R	R[Rdest] = R[R1] & R[R2]
0011	OR	R	R[Rdest] = R[R1] | R[R2]
0100	SUB	R	R[Rdest] = R[R1] - R[R2]
0101	NEG	R	R[Rdest] = -R[R1]
0110	NOT	R	R[Rdest] =~R[R1]
0111	CPY	R	R[Rdest] = -R[R1]
1000	LRG	I	R[R1]= Imediato
1001	BLT	I	PC=(R[R1][7]==1? PC+Imediato : PC+1)
1010	BGT	I	PC=(R[R1][7]==0? PC+Imediato : PC+1)
1011	BEQ	I	PC=(R[R1]==0? PC+Imediato : PC+1)
1100	BNE	I	PC=(R[R1]!=0? PC+Imediato : PC+1)
1101	JMP	J	PC=Imediato
1110	INPUT	R	R[Rdest] = SW[7:0]
1111	OUTPUT	R	HEX[Rdest]=R[R1]

Tabela 1: Instuções do processador Nano

Arquitetura interna

A Figura a seguir traz um modelo esquemático da arquitetura interna do Nano.



CmdULA	Operação
0	R1
1	R1 + R2
2	R1 & R2
3	R1 | R2
4	R1 - R2
5	- R1
6	~R1

Figura 2: Arquitetura interna do processador Nano

Como principais elementos internos do Nano, destacam-se os seguintes:

- Unidade Lógica e Aritmética – ULA, a qual é responsável por efetuar todas as operações lógicas e aritméticas necessárias a execução de todas as instruções do processador;
- Banco de Registradores, o qual serve como memória de trabalho do processador;
- Unidade de Controle – CTRL, a qual é responsável por buscar, decodificar e controlar a execução das instruções do processador;
- Memória de instruções, espaço de memória onde são carregados os programas para serem executados
- O registrador do contador de programa, o registrador PC, o qual armazena o endereço da instrução a ser buscada e executada
- Estrutura de execução das instruções de desvio condicional e incondicional, resposável por controlar modificar o conteúdo do registrador PC quando da execução de uma instrução de desvio condicional ou incondicional

Observa-se também que estes elementos internos são interligados por sinais de controle e de transferência de informações, dentre os quais se destacam os seguintes:

- **OP**: Palavra com os 4 bits mais significativos da instrução armazenada na memória. É com base no seu conteúdo que a unidade de controle identifica a instrução a ser executada.
- **End.JMP** e **Dsv**: Contém o conteúdo do campo imediato das instruções de desvio condicional e incondicional. Indica em quantas instruções a execução do processador deverá ser desviada.
- **SelDsv**: Sinal que indica que deve ser executado um desvio condicional, somando ao conteúdo do PC o conteúdo do campo imediato.
- **SelJMP**: Sinal que indica que deve ser executado um desvio incondicional, carregando o PC o com conteúdo do campo imediato.
- **LdPC**: Sinal que indica que o conteúdo do registrador PC deve ser atualizado.
- **EndWR**: Palavra binária extraída da instrução do processador com o endereço do registrador que deve receber o resultado da operação efetuada pela ULA
- **EndR1**:Palavra binária extraída da instrução do processador com o endereço do registrador que contem o primeiro operando das operações tipo R e I, com exceção da instrução LRG, na qual ela indica o endereço do registrador destino.
- **EndR2**: Palavra binária extraída da instrução do processador com o endereço do registrador que contem o segundo operando das operações tipo R .
- **Imediato**: Palavra binária extraída da instrução do processador com o campo imediato das instruções tipo I e tipo J.
- **ResultULA**: Palavra binária com o resultado da operação da ULA.
- **SelDtWr**: Sinal que seleciona o conteúdo a ser gravado no banco de registradores,

indicando se o mesmo será proveniente da ULA ou se será o campo imediato da instrução.

- **Wr**: Sinal que habilita a escrita no bando de registradores.
- **SelRegWr**: Sinal que seleciona se o endereço do registrador a ser gravado será proveniente do campo RegWr ou do campo R1 da instrução.
- **CmdULA**: Palavra binária contendo o comando para a ULA.
- **LdPC**: Sinal que sincroniza a carga do registrador PC.
- **LdOUTPUT**: Sinal que sincroniza a carga do registrador que implementa a porta de saída do processador.

Configuração das chaves, botões e displays da placa DE-2 no projeto do processador

A figura a seguir indica como alguns recursos da placa DE-2 foram alocados no projeto do processador.

Os botões KEY0 e KEY3 foram utilizados respectivamente como botão de reset e botão de inserção de clock manual para o procesador.

A chave SW17 foi utilizada para configurar o modo de operação do processador. Quando ligada, com a alavanca voltada para cima, o processador funciona automaticamente com gerador de clock da placa. Quando desligada o processador entra no modo passo a passo, tendo o seu clock fornecido pelo botão KEY3.

A chave SW16 foi utilizada para configurar o modo de operação dos displays de sete seguimentos. Quando esta chave está ligada, com a alavanca voltada para cima, os displays são utilizados para inspecionar o funcionamento do processador. Nesta condição o display HEX7 mostra o estado da máquina de estado do processador. Os displays HEX5 e HEX4 mostram o conteúdo do contador de programa (PC). Os displays HEX3, HEX2, HEX1 e HEX0 são utilizados para mostrar o centeúdo da posição de memória apontada por PC, ou seja, a instrução que foi lida da memória.

Quando a chave SW16 estiver desligada, com a alavanca voltada para baixo, os displays serão utilizados para visualizar as portas de saída do processador, acessadas por meio da instrução OUTPUT. Os displays HEX0 e HEX1 visualizam a porta zero, os displays HEX2 e HEX3 a porta um, os diaplays HEX4 e HEX5 a porta dois e os displays HEX6 e HEX7 a porta três.

Deve-se observar que as infomações mostradas nos displays só são válidas no momento determinado pela programaçao da máquina de estados do processador.

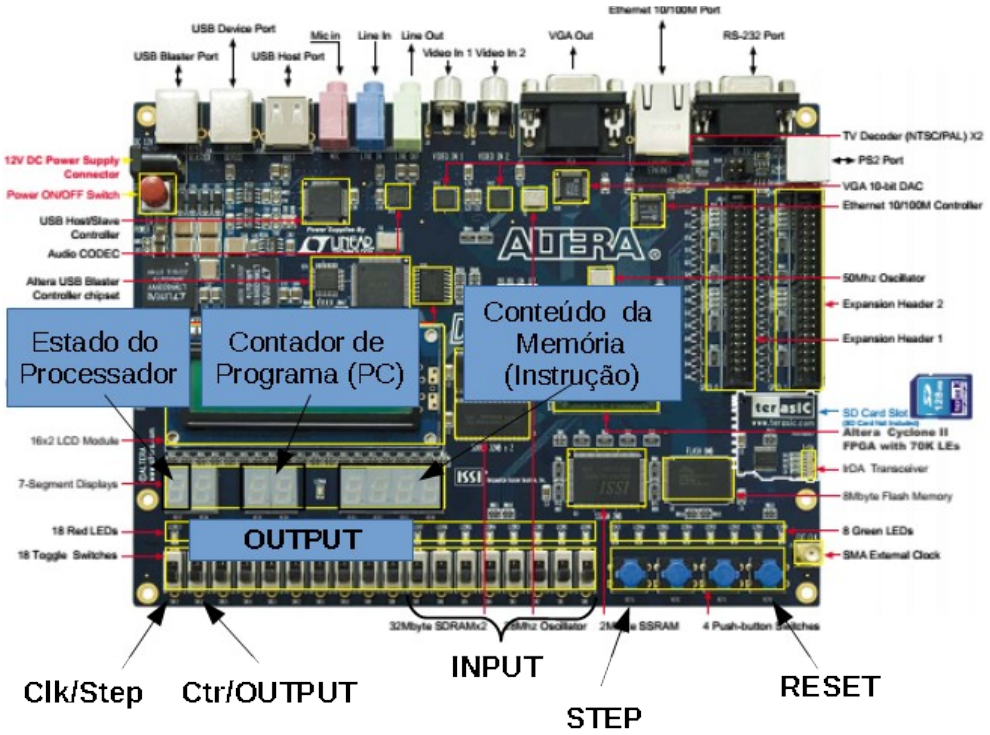


Figura 3: Alocação dos recursos da placa DE2-70 no projeto do procesador

Orientações gerais para utilização do template do processador

O projeto do processador foi fornecido com todos os módulos necessários a sua implementação, desta forma o seu trabalho será substituir e/ou modificar estes módulos de forma a atender as suas especificações de funcionamento. A tabela a seguir traz uma pequena descrição de cada um destes módulos.

Módulo	Descrição
divClk	Módulo utilizado para reduzir a frequência de operação do sinal de clock da placa DE2-70
debounce	Módulo utilizado para remover o “bounce” das teclas
pulse	Módulo utilizado para gerar um pulso de duração pré-determinada ao ser pressionada a tecla a ele associada
topPlaca	Módulo utilizado para permitir o mapeamento do processador nos recursos da placa DE2-70
processador	Módulo que implementa o processador
rom	Módulo que implementa a memória de programa do processador
muxes	Módulo que implementa os multiplexadores utilizados para controlar o caminho de dados e a lógica de carga do registrador PC
ctrl	Módulo que implementa a unidade de controle do processador
ula	Módulo que implementa a unidade lógica e aritmética processador
BancoRegistra-dores	Módulo que implementa o banco de registradores do processador
Registraodor	Módulo que implementa o registrador utilizados para representar o registrador PC e a porta de saída do processador
display7seg	Módulo que implementa a interface de conexão com os displays da placa DE2-70. Este módulo efetua a conversão do padrão BCD8421 para sete seguimentos.
Lrg.mif	Arquivo de teste das instruções LRG e OUTPUT
somador	Módulo que implementa o somador de 8 bits utilizado para incrementar o PC e para calcular o endereço de desvio das instruções de desvio condicional
Processador.vwf	Arquivo de simulação do projeto

Desta forma, quando o seu objetivo for simular e debugar o seu projeto utilizando o arquivo waveform, configure o módulo processador como módulo top da sua hierarquia de projeto. Quando desejar fazer a síntese para carga na placa DE2-70, configure o projeto para utilizar o módulo topPlaca como top da sua hierarquia de projeto.

A Figura 4 a seguir apresenta o resultado esperado para a simulação do processador com as instruções contidas no arquivo Lrg.mif.

Em destaque nesta figura temos:

1. Em 1 e 2, a introdução do pulso de reset e a consequente reinicialização do processador;
2. O processo de escrita no banco de registradores como consequência da execução das instruções LRG
3. O processo de escrita na porta de saída do processador como consequência da execução das instruções OUTPUT

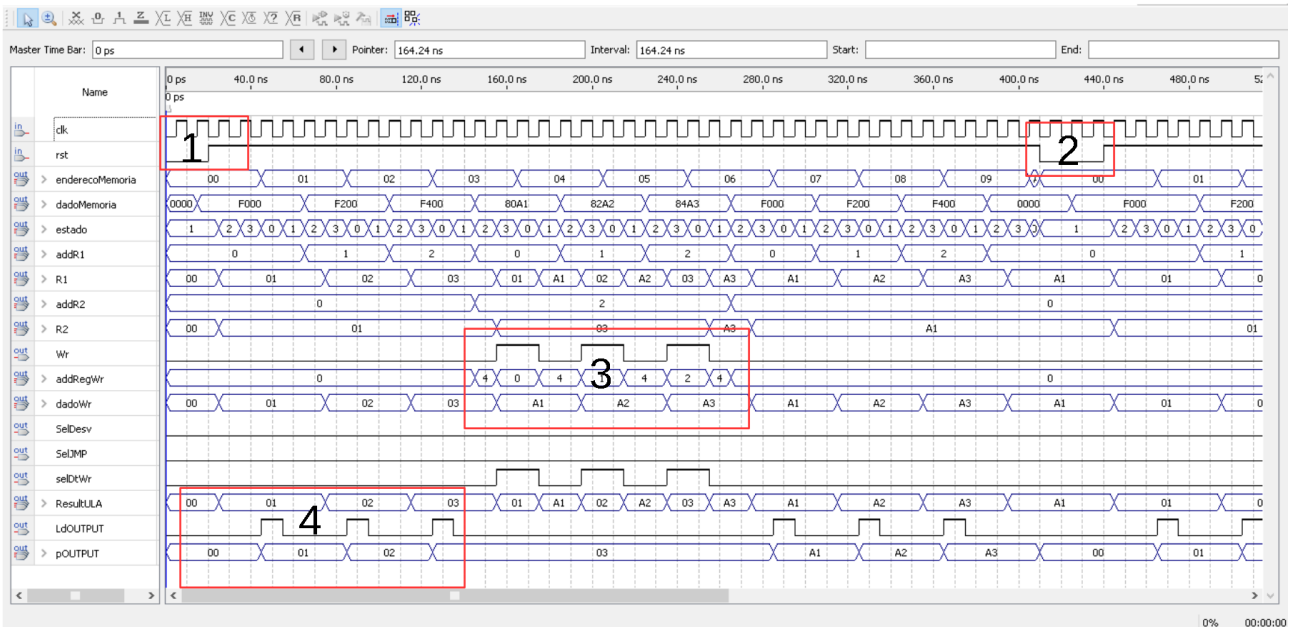


Figura 4: Simulação do processador com as instruções do arquivo Lrg.mif

Passo a passo para testar o projeto do processador

Para cada etapa dos testes do processador recomenda-se primeiramente efetuar a simulação por meio do waveform, utilizando para tanto o módulo processador como top do projeto, e, só após verificar que está tudo ok, fazer a síntese final substituindo o top do projeto pelo módulo topPlaca para efetuar a carga na placa.

1. Preencha ou substitua o conteúdo dos arquivos com os projetos que foram desenvolvidos por sua equipe.
2. Recompile o projeto no Quartus.
3. Certifique-se que as chaves SW17 e SW16 estejam ligadas, com as alavancas voltadas para cima.
4. Baixe o arquivo sof na placa e verifique o seu funcionamento.
    1. O led LEDG0 deve permenacer piscando, enquanto os leds LEDG1 e 2 devem permanecer acessos.
5. Coloque o processador para executar no modo passo a passo. Ao fazer isto, apesar de o led LEDG0 continuar piscando, os displays permanecerão estáticos, indicando que o processador está parado. Pressione o botão KEY3. Cada vez que este botão é pressionado é gerado um pulso de clock para o processador, o que pode ser verificado por meio do led LEDG1, que pisca juntamente com a geração deste pulso, e também por meio dos displays, que modificam os seus estados juntamente com a evolução dos estados internos do processador.
6. Pressione o botão KEY0, verifique que o led LEDG2 pisca, indicando que foi introduzido um pulso de reset no processador. Neste momento, tanto o estado do processador quanto o contador de programa voltam ao estado zero.
7. Faça um programa que carregue o conteúdo dos quatro primeiros registradores do banco de registradores nas portas de saída. Após carregar o conteúdo dos registradores nos displays, o programa deve ficar em loop por meio de uma instrução JMP -1. Desligue a chave SW16 para poder visualizar o conteúdo das portas de saída nos diplays da placa.
8. Faça um programa que carregue o registrador oito com a soma do conteúdo dos sete outros registradores. Carregue o resultado desta soma na porta de saída zero.
9. Faça um programa que carregue o conteúdo da porta de entrada no registrador zero. Em seguida carregue o conteúdo do registrador zero na porta de saída um.
10. Faça um programa que carregue os quatro primeiro registradores com os valores 0x10, 0x11, 0x12 e 0x13. Carregue o valor de cada um deste registradores nas portas de saída do processador.
11. Implemente e teste as operações SUB, AND, OR, NEG, NOT e CPY.
12. Implemente e teste as operações BGT, BEQ e BNE.

Lembre-se que a cada modificação em qualquer dos arquivos do projeto, este deve ser recompilado e carregado novamente na placa.

Instruções para uso do arquivo instruções.odg ou instruções.xls.

A planilha contida nos arquivos intruções.odg e instruções.xls tem como objetivo simplificar a geração do conteúdo a ser carregdo no arquivo processador.mif.

Esta planilha possui três tabelas que permitem gerar automaticamente o código de máquina das 16 instruções disponíveis em nosso processador. Cada tabela deve ser utilizada para gerar o código de máquina de um tipo específico de instrução.

Por ter sido baseado na arquitetura do processador MIPS, o nosso processador buscou também seguir o princípio de regularidade e simplicidade que norteiam o projeto dos processadores RISC.

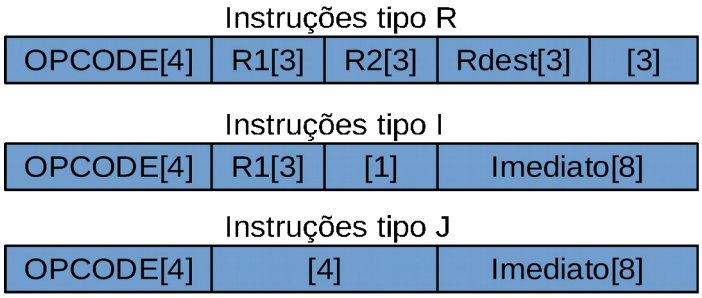


Figura 5: Formato das instruções do processador

Desta forma, as instruções do processador foram organizadas em três grupos distintos, denominados Instruções Tipo R, Instruções Tipo I e instruções Tipo J. A figura 4 a seguir apresenta como os 16 bits do código de máquina de cada instrução de cada um destes grupos de instruções estão divididos.

As instruções tipo R são aquelas em que todo a informação necessária à execução da instrução já se encontra carregada no banco de registradores do processador. Fazem parte deste grupo as instruções ADD, AND, OR, SUB, NEG, NOT, CPY, INPUT e OUTPUT.

As instruções tipo I são aquelas em que parte da informação necessária à sua execução é fornecida no próprio código de máquina da instrução. Estas instruções possuem no seu código de máquina um campo formado pelos 8 bits menos significativos denominado de campo imediato. Desta forma, o processamento destas instruções envolverá tanto o conteúdo de um registrador quanto o conteúdo do campo imediato. Fazem parte deste grupo de instruções as instruções LRG, BNE, BEQ, BLT e BGT.

Por último, o grupo de instruções tipo J possui aenas uma instrução, a instrução JMP. Esta instrução possui apenas o opcode da instrução e o campo imediato.

A figura 7 a seguir apresenta como a planilha de geração dos códigos de máquina de cada uma destas instruções está estruturada. Como se pode observar, abaixo de cada tabelaa que permite gerar o código de máquina das instruções existe uma segunda tabela que apresenta de forma sucinta o funcionamento de cada uma das instruções.

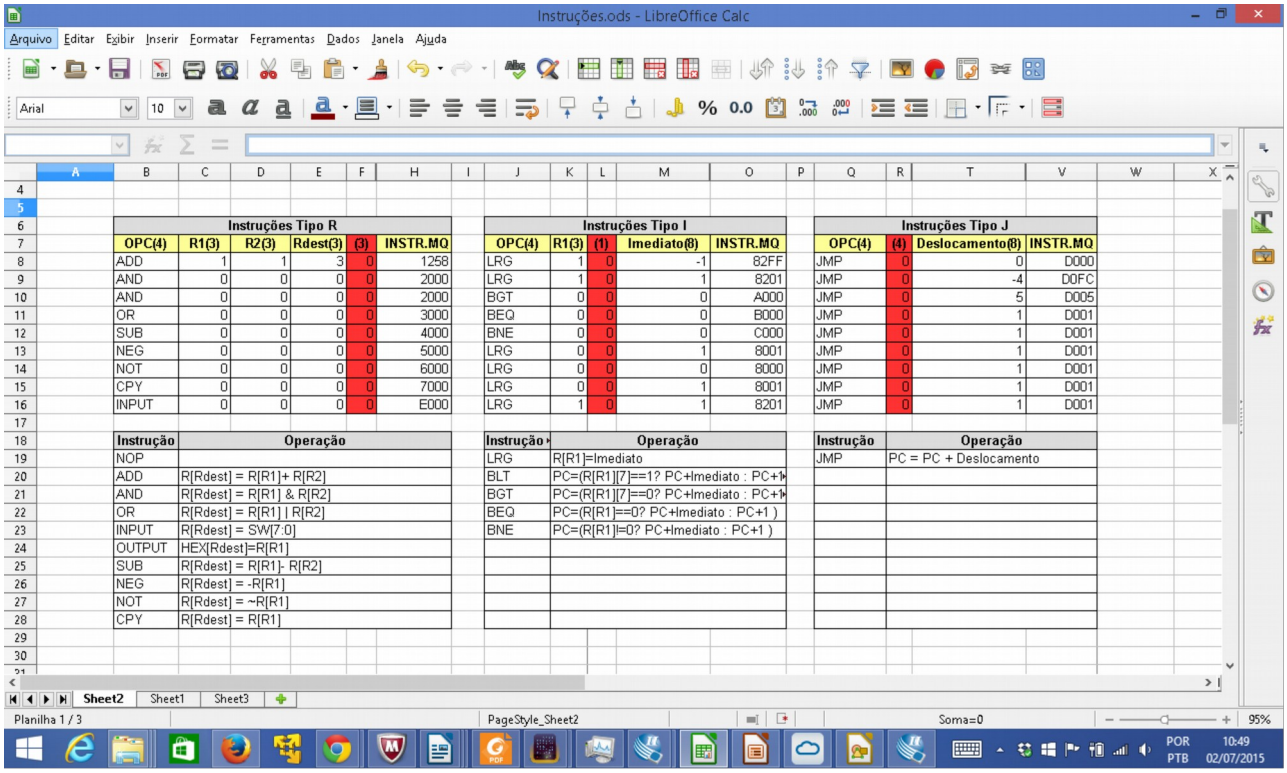


Figura 6: Template para geração dos códigos de máquina do processador por meio do uso dos arquivos instruções.odg e instruções.xls

Para a geração do código de máquina das instruções basta preencher os campos correspondentes necessários para definir a instrução desejada para obter no campo INSTR.MQ. o código hexadecimal da instrução.

Os códigos de máquina assim obtidos devem ser então copiados para o arquivo mif para que possam ser introduzidos na memória do processador durante o seu processo de síntese.