GABARITO AD1 - Organização de Computadores 2014.2

1) (1,0) Faça uma pesquisa no livro "Arquitetura e Organização de Computadores" de William Stallings e descreva os métodos utilizados para lidar com desvios condicionais em arquiteturas que utilizam pipeline.

O Desvio condicional consiste no principal impedimento para um fluxo constante de instruções em um pipeline de instruções. Diversas abordagens são adotadas para lidar com o desvio condicional, entre elas:

Múltiplos fluxos: *Consiste em duplicar os estágios iniciais da pipeline para permitir a busca de ambas as instruções, usando assim dois fluxos de instruções. Esta abordagem apresenta problemas como o atraso à contenção de acesso a registradores e à memória, além da entrada de instruções adicionais na pipeline antes que seja tomada a decisão sobre o desvio original.*

Busca antecipada da instrução-alvo de desvio*. Consiste em buscar antecipadamente tanto a instrução-alvo de desvio quanto a instrução consecutiva ao desvio, no instante em que uma instrução de desvio-condicional é reconhecida. A instrução-alvo é armazenada em um registrador, até que a instrução de desvio seja executada. Seja o desvio tomado ou não, a próxima instrução a ser executada terá sido buscada antecipadamente.*

Memória de laço de repetição: *Consiste em usar uma pequena memória de alta velocidade, mantida pelo estágio de busca de instrução da pipeline, que é usada para manter as* ***n*** *instruções buscadas mais recentemente, em seqüência. Na ocorrência do desvio, será verificado se a instrução-alvo está presente nesta memória.*

Previsão de desvio*: Consiste em várias técnicas que podem ser usadas para prever se um desvio será tomado ou não, entre as mais comuns estão as seguintes.*

1) *Prever que o desvio nunca será tomado: esta previsão não depende do histórico de execução de instruções até o momento, não supõe que o desvio não seja tomado e continua a buscar instruções na seqüência em que ocorrem no programa.*
2) *Prever que o desvio sempre será tomado: conforme a anterior diferindo que o desvio será tomado, buscando sempre as próximas instruções a partir do endereço-alvo do desvio.*
3) *Prever se o desvio será tomado ou não conforme o código de operação: em casos de determinados códigos de operação, o processador supõe que o desvio sempre é tomado; em outros, que o desvio nunca é tomado, estudo feito com esta técnica apresenta taxas de sucesso superiores a 75%.*
4) *Prever o desvio com base em chaves de desvio tomado e de desvio não tomado: consiste em uma estratégia dinâmica onde um ou mais bits podem estar associados a cada instrução de desvio condicional, usados para refletir a história recente da execução da instrução. Bits estes chamados de chaves de desvio tomado ou de desvio tomado ou de desvio não tomado.*
5) *Prever o desvio com base em uma tabela de histórico de desvios: Esta técnica mantém uma pequena tabela de histórico de instruções de desvio executadas mais recentemente, incluindo um ou mais bits de entrada. O processador pode endereçar essa tabela de maneira associativa, tal como uma memória cache, ou usar bits de mais baixa ordem do endereço das instruções de desvio. A tabela de histórico de desvios é uma pequena memória cache associada ao estágio de busca de instrução da pipeline.*

Atraso de desvio*: técnica para reordenar automaticamente as instruções de um programa, de modo que instruções de desvio ocorram mais tarde do que ocorrem de fato na seqüência especificada.*

2) (1,0) Considere a máquina apresentada na aula 4. Descreva detalhadamente (do mesmo modo que é apresentado na aula 4) como é realizada a execução das seguintes instruções:

a) LDA 200

1) RI ← (CI) , ou seja, RI ← recebe a Instrução contida no endereço contido no CI
2) CI ← CI + 1
3) Decodificação do código de operação
- recebe os bits do código de operação
- produz sinais para a execução da operação de leitura em memória
4) Execução da operação
- A UC emite sinais para que o valor do campo operando (200) seja transferido para a REM
*- A UC emite sinais para que o valor contido no REM seja transferido para o barram*ento de endereços
- A UC ativa a linha READ do barramento de controle
- Conteúdo da posição da memória, conforme endereço contido no barramento de endereços (200), é transferido através do barramento de dados para o RDM
- O conteúdo do RDM é transferido para o registrador acumulador (ACC <- RDM)

b) SUB 22

1) RI ← (CI) , ou seja, RI ← recebe a instrução contida no endereço contido no CI
2) CI ← CI + 1
3) Decodificação do código de operação
- recebe os bits do código de operação
- produz sinais para a execução da operação de subtração
4) Execução da operação
- A UC emite sinais para que o valor do campo operando (22) seja transferido para a REM
*- A UC emite sinais para que o valor contido no REM seja transferido para o barram*ento de endereços
- A UC ativa a linha READ do barramento de controle
- Conteúdo da posição da memória, contido no barramento de endereços (22), é transferido através do barramento de dados para o RDM
- UC emite sinais para transferir conteúdo do acumulador para UAL, (UAL ← ACC) - 1o. termo
- O conteúdo do RDM é transferido para o registrador acumulador (ACC ← RDM)
- UC emite sinais para transferir conteúdo do acumulador para UAL. (UAL ← ACC) - 2o. termo
- A UC emite sinais para a UAL executar da operação se subtração
- A UC emite sinais para a UAL liberar resultado e armazenar no acumulador (ACC ← UAL)

c) JP 300

1) RI ← (CI) , ou seja, RI ← recebe a Instrução contida no endereço contido no CI
2) CI ← CI + 1
3) Decodificação do código de operação
- recebe os bits do código de operação
- produz sinais para a execução da operação de salto condicional
4) Execução da operação
- UC emite sinal para transferir conteúdo do acumulador para UAL
- UAL ← ACC
- Executa operação de comparação
- Se Resultado = verdadeiro, isto é, ACC > 0, CI ← 300

3) (1,5) Considere uma máquina com 1G células de memória onde cada célula armazena uma palavra e cada instrução tem o tamanho de uma palavra. Esta máquina possui um conjunto de instruções com 16 instruções distintas, sendo cada uma delas composta de um código de operação e dois operandos, que indicam o endereço de memória.

a) Qual o tamanho mínimo do REM?

REM = Barramento de endereços, este terá a capacidade de endereçar 1G células = N
N = 1G células => $N = 2^{30}$ => e = 30 bits
REM = barramento de endereços = 30 bits

b) Qual o tamanho mínimo do RI?

O tamanho de RI deverá ser o tamanho de uma instrução
Tamanho de uma instrução = código de operação (total de 16) + 2 operandos (endereço de memória)
Tamanho de uma instrução = 4 bits + 2 x 30bits = 64 bits

Tamanho mínimo do RI = tamanho de uma instrução = 64 bits

c) Qual o tamanho mínimo do RDM?

O tamanho mínimo do RDM deverá ser o tamanho de uma célula
O tamanho de uma célula = tamanho de uma palavra = tamanho de uma instrução
Então, RDM = tamanho de uma instrução => ***RDM = 64 bits***

d) Qual o tamanho da memória em bits?

T = M x N => T = tamanho da célula x memória principal =>
tamanho da célula (M) = 64 bits
total de endereços da memória (N) = 2^{30} células
T = 64 bits/célula x 2^{30} células => ***T = 2^{36} bits ou T = 64 Gbits***

e) Se a largura do barramento de dados desta máquina for igual à metade do tamanho de uma instrução, como funcionará o ciclo de busca?

Seriam necessários 2 ciclos de busca para transferir uma instrução completa.

4) (1,5) Considere o sistema visto na aula 4 que possui uma memória com 256 células, sendo que cada célula pode armazenar 12 bits. Cada instrução possui 12 bits, sendo que 4 bits indicam o código de operação e 8 bits indicam o operando. Suponha que, em um determinado momento, alguns endereços da memória contenham os seguintes conteúdos (todos os valores estão em hexadecimal): Considere que nos endereços 20 a 27 estão armazenadas instruções de um programa.

a) Traduza as instruções para as siglas e operandos correspondentes. Por exemplo, a instrução contida no endereço 20 é LDA 50.

Endereço	*Instrução*		*Descrição*
20	*150*	*LDA 50*	*ACC ← (50)*
21	*351*	*ADD 51*	*ACC ← ACC + (51)*
22	*260*	*STR 60*	*(60) ← ACC*
23	*152*	*LDA 52*	*ACC ← (52)*
24	*460*	*SUB 60*	*ACC ← ACC - (60)*
25	*527*	*JZ 27*	*Se ACC = 0, CI ← 27*
26	*261*	*STR 61*	*(61) ← ACC*
27	*000*	*HALT*	*Parar a execução*

b) Supondo que o CI possua o endereço 20, indique como será realizada a execução de cada instrução deste programa e mostre o conteúdo dos registradores RDM, REM, RI, CI e ACC e das células de memória cujos endereços são 50, 51, 52, 60 e 61 ao final da execução deste programa, para cada um dos casos abaixo:

i) Inicialmente os conteúdos das células 50, 51, 52, 60 e 61 são 010, 015, 025, 030 , 050

CI	*RI*	*Descrição da Instrução*	*RDM*	*REM*	*ACC*	*50*	*51*	*52*	*60*	*61*
20	*x*	*x*	*x*	*x*	*x*	*010*	*015*	*025*	*030*	*050*
21	*150*	*ACC ← (50)*	*010*	*050*	*010*	*010*	*015*	*025*	*030*	*050*
22	*351*	*ACC ← ACC + (51)*	*015*	*051*	*025*	*010*	*015*	*025*	*030*	*050*
23	*260*	*(60) ← ACC*	*025*	*060*	*025*	*010*	*015*	*025*	*025*	*050*
24	*152*	*ACC ← (52)*	*025*	*052*	*025*	*010*	*015*	*025*	*025*	*050*
25	*460*	*ACC ← ACC - (60)*	*025*	*060*	*000*	*010*	*015*	*025*	*025*	*050*
27	*527*	*Se ACC = 0, CI ← 27*	*527*	*026*	*000*	*010*	*015*	*025*	*025*	*050*
*	*000*	*HALT*	*000*	*027*	*000*	*010*	*015*	*025*	*025*	*050*

ii) Inicialmente os conteúdos das células 50, 51, 52, 60 e 61 são 010, 015, 035, 040, 050

CI	*RI*	*Descrição da Instrução*	*RDM*	*REM*	*ACC*	*50*	*51*	*52*	*60*	*61*
20	*x*	*x*	*x*	*x*	*x*	*010*	*015*	*035*	*040*	*050*
21	*150*	*ACC ← (50)*	*010*	*050*	*010*	*010*	*015*	*035*	*040*	*050*
22	*351*	*ACC ← ACC + (51)*	*015*	*051*	*025*	*010*	*015*	*035*	*040*	*050*
23	*260*	*(60) ← ACC*	*025*	*060*	*025*	*010*	*015*	*035*	*025*	*050*
24	*152*	*ACC ← (52)*	*035*	*052*	*035*	*010*	*015*	*035*	*025*	*050*
25	*460*	*ACC ← ACC - (60)*	*025*	*060*	*010*	*010*	*015*	*035*	*025*	*050*
26	*527*	*Se ACC = 0, CI ← 27*	*527*	*025*	*010*	*010*	*015*	*035*	*025*	*050*
27	*261*	*(61) ← ACC*	*010*	*061*	*000*	*010*	*015*	*035*	*025*	*010*
*	*000*	*HALT*	*000*	*027*	*000*	*010*	*015*	*035*	*025*	*010*

5) (1,0) Explique o que são microinstruções horizontais e verticais utilizadas em arquiteturas com unidade de controle microprogramada.

Em uma arquitetura microprogramada, a unidade de controle é especificada por um microprograma que consiste de uma seqüência de instruções de uma linguagem de microprogramação. Estas instruções são muito simples e especificam microoperações. Uma unidade de controle microprogramada é implementada com circuitos lógicos e é capaz de seguir uma seqüência de microinstruções gerando sinais de controle para que cada uma delas seja executada. Os sinais de controle gerados por uma microinstrução são usados para causar transferências de dados entre registradores e memória e execução de operações pela ULA.

As microinstruções horizontais tem como característica ter funções distintas para cada bit que a compõe, como por exemplo, controlar uma linha de controle interna da UCP, controlar uma linha de barramento externo de controle, definir condição de desviro e endereço de desvio entre outras. Tem a vantagem de ser simples e direto possível, podendo controlar várias microoperações em paralelo, além de uma eficiente utilização do hardware. E possui a desvantagem de maior ocupação de espaço de memória de controle em relação à microinstrução vertical.

As microinstruções verticais se caracterizam por possuir um decodificador extra para identificar quais as linhas que serão efetivamente ativadas. Sua principal vantagem é reduzir o custo da Unidade de controle em função do menor tamanho da instrução, o que poderá ser necessário uma maior quantidade de instruções. Tem como principal desvantagem a redução do tempo devido à necessidade da decodificação dos campos de cada microinstrução.

6) (1,5) Considere uma máquina que possa endereçar 2 Giga bytes de memória física, sendo que cada endereço referencia uma célula de 8 bytes. Ela possui uma memória cache que pode armazenar 2 Mega blocos, sendo um bloco por linha e cada bloco possui 4 células. Mostre o formato da memória cache, indicando os campos necessários (tag, bloco) e o número de bits para cada campo, o formato de um endereço da memória principal, indicando os bits que referenciam os campos da cache, e a capacidade em bits que a memória cache deve possuir (pode deixar a conta indicada) para os seguintes mapeamentos:

a) Mapeamento direto.

M*emória Principal*

⇒ *Tamanho da memória (em bytes) = 2Gbytes, como cada célula contém 8 bytes, temos, então, N = 256M células*

⇒ *A MP está organizada em blocos de 4 células, (K = 4 células/bloco)*

N = 256M células e K = 4 células / bloco, o total de blocos da MP (B) será:

Total de blocos: B = N / K => B = 256M células / 4 células/bloco => B = 64 M blocos

Memória Cache

O*BS: O K (quantidade de células/bloco) tem de ser igual a MP.*

=> Tamanho da memória cache (em blocos ou linhas) => Q = 2M blocos

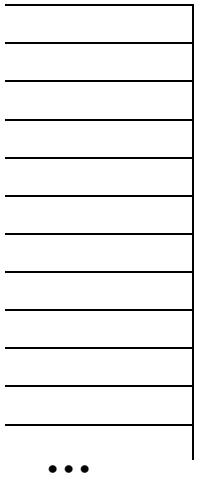
=> Tamanho da memória cachê em células = Q x K = 2M blocos x 4 células/blocos

= 8M células (64 Mbytes)

Memória principal

256M células: N

64M blocos: B

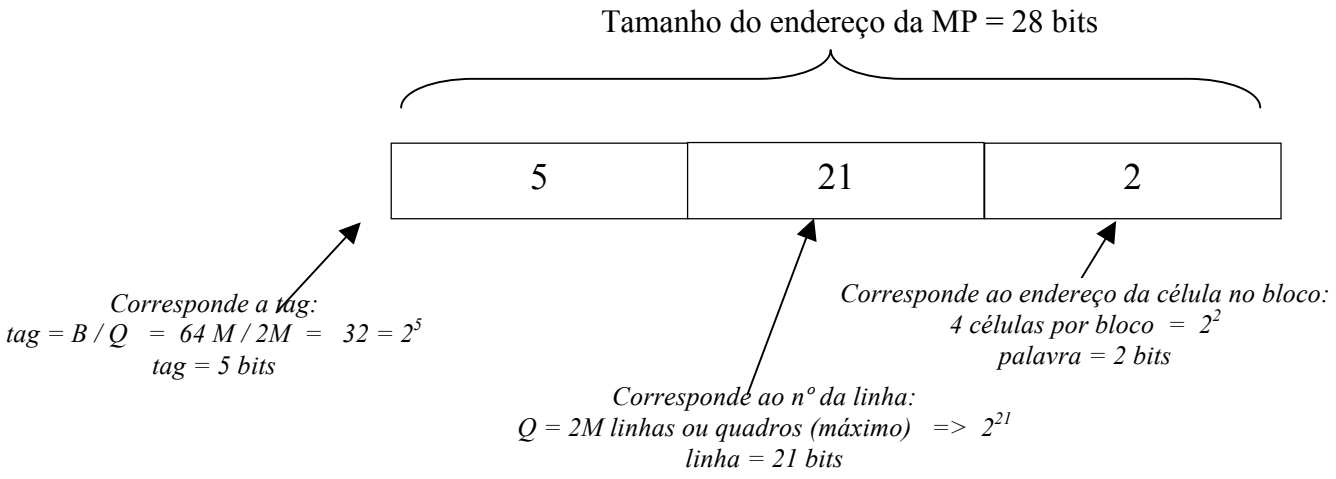


Organização da cache

linha	válido	tag	Conteúdo (bloco)
0	1 bit	5 bits	4 células de 64 bits cada = 256bits
1			
2			
3			
4			
5			
.....			
Q - 2			
Q - 1			

Para endereçarmos toda a MP precisamos da seguinte quantidade de bits (E)

sendo $N = 2^E$ => N = 256M células => $N = 2^{28}$ => E = 28 bits

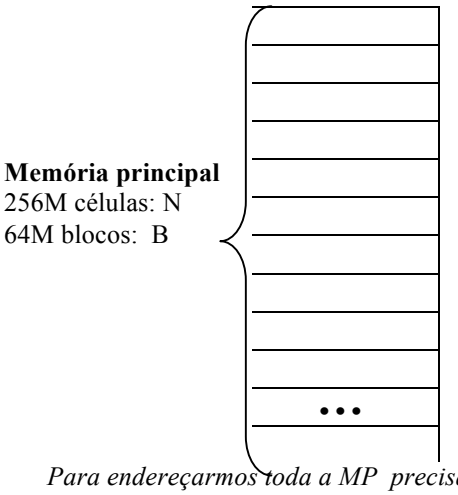


b) Mapeamento totalmente associativo.

Memória Principal
=> N = 256M células
=> K = 4
=> B = 64 M blocos

Memória Cache
OBS: O K (quantidade de células/bloco) tem de ser igual a MP.
=> Q = 2M blocos
=> Tamanho da memória cache = 64 Mbytes

Organização da cache

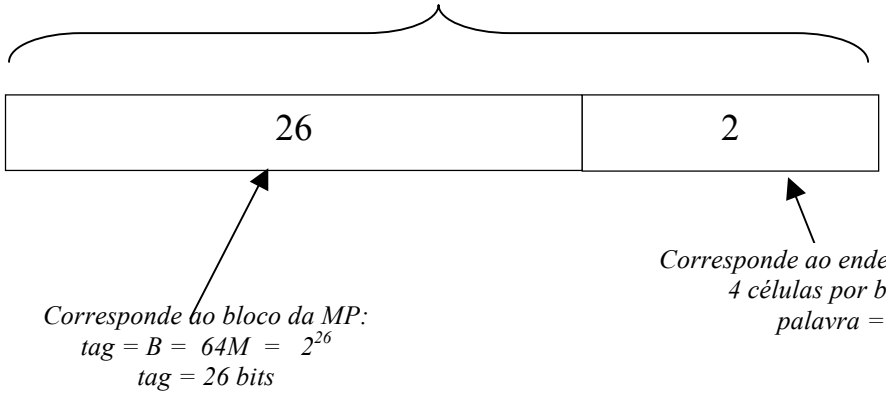


linha	válido	tag	Conteúdo (bloco)
0	1 bit	26 bits	4 células de 64 bits cada = 256bits
1			
2			
3			
4			
5			
.....			
Q - 2			
Q - 1			

Para endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 28 bits

Como o bloco pode ser alocado em qualquer posição da memória cacha a tag indicará qual dos blocos da MP está alocado naquela posição da memória cachê

Tamanho do endereço da MP = 28 bits

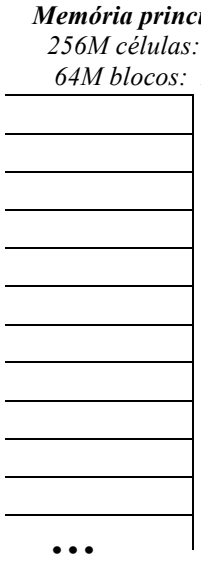


c) Mapeamento associativo por conjunto, onde cada conjunto possui duas linhas, cada uma de um bloco.

Memória Principal
=> K = 4 (quantidade de células/bloco)
=> B = 64M blocos
=> *N = 256M células*

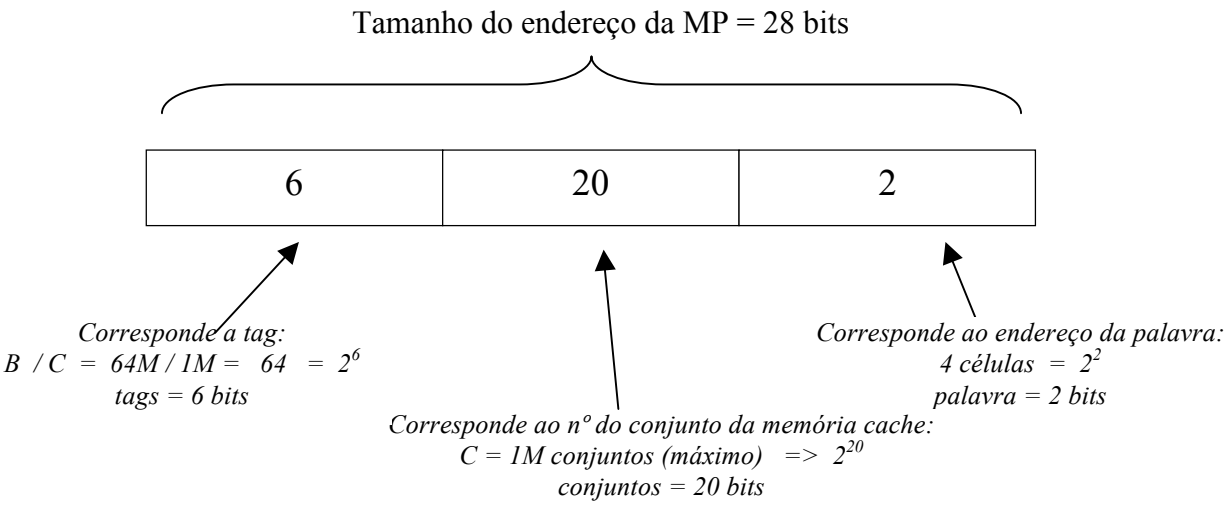
Memória Cache
O*BS: O K (quantidade de células/bloco) tem de ser igual ao da MP.*

=> Q = 2M blocos
=> Tamanho da memória cache = 64 Mbytes
=> 1 conjunto = 2 linhas (ou quadros) => Total de conjuntos (C) = 2M células / 2 => C = 1M conjuntos



Conjunto	linha	válido	tag	Conteúdo (bloco)
0	0	1 bit	6 bits	4 células de 64 bits cada = 256bits
	1			
1	2			
1	3			
2	4			
	5			
.....				
C - 1	Q - 2			

P*ara endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 28 bits*



7) (1,5) Faça uma pesquisa e indique um microprocessador comercial, cuja unidade de controle possa ser caracterizada como sendo por hardware e um outro, cuja unidade de controle possa ser caracterizada por microprogramada. Explique detalhadamente as suas indicações e coloque a fonte de sua pesquisa.

Fonte: livro texto da disciplina

As unidades de controle microprogramadas estão presentes nos processadores da arquiteturas CISC Esta arquiteturas tem como exemplo: processadores da série Intel Ix (I3,I5 e I7). A unidade de controle microprogramada é utilizada para se desenvolver a implementação de complexas instruções que não podem ser implementadas em forma de hardware. Isto torna mais fácil de se projetar e flexibilizar uma unidade de controle. A microprogramação tem a vantagem de permitir até a emulação de outro computador, e ainda, que uma instrução possa ser desenvolvida e ser utilizada em diferentes modelos de hardware.

As unidades de controle por hardware estão presentes nos processadores das arquiteturas RISC. Esta arquitetura tem como exemplo: a serie Power PC da Motorla/IBM. Nesta unidade de controle, as microinstruções serão executadas diretamente pelo hardware. Em geral, o número de equações booleanas pode ser muito grande, o que pode tornar difícil a implementação combinatória de um grande número de instruções. A vantagem da implementação por hardware sobre microprogramação está na velocidade de execução das instruções. Esta velocidade de execução pode compensar o baixo número de instruções, o que viabiliza este tipo de arquitetura.

8) (1,0) Para os mesmo microprocessadores escolhidos na questão anterior explique a hierarquia de memória dessas máquinas.

Organização da memória do Intel Ix (I3, I5 e I7)

Fonte base para o texto: http://www.clubedohardware.com.br/artigos/Por-Dentro-da-Microarquitetura-Intel-Sandy-Bridge/2146/1

1º. Nível: Registradores presentes apenas no interior do núcleo das CPUs. Os principais registradores da série Ix são:

Registradores de uso geral (32bits) EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
Registradores de segmento (16 bits): CS, DS, SS, ES, FS, GS
Registrador de controle e status do programa (32 bits): EFLAGS
Ponteiro de instrução (32bits): EIP
Registradores de gerenciamento de memória: Global Descriptor Table Register (GDTR – 64 bites), Local Descriptor Table Register (LDTR), Interrupt Descritor Table Register: (IDTR - 64 bits), Task Register: (TR - 64 bits)
Registradores de controle: CR0, CR1, CR2, CR3, CR4 e CR8 (64 bits)
Registrador de controle extendido: XCR0 (64 bits)

2º. Nível: Memória Cache divididas nos subníveis L1, L2 e L3 presentes dentro do chip da CPU. Os processadores I3, I5 e I7 da 2ª. geração utilizam a microarquitetura Sandy Bridge. Esta arquitetura possui uma cache de instruções decodificadas que poderia ser denominada como uma cache L0, capaz de armazenar cerca de 1536 instruções, em torno de 6KB. A cache L1 não diferencia da microarquitetura da 1ª. Geração (Nehalem), é dividida em 2: cache de instruções com cerca de 32KB, e cache de dados de igual tamanho. A cache L2 é uma cache intermediária com 256KB por núcleo do chip do processador. A cache L3 é compartilhada entre os núcleos, ou seja, não é ligada a um núcleo em particular. A cache L3 pode ser utilizadas pelo componente gráfico presente no chip do processador para armazenar dados, em especial texturas aumentando o desempenho 3D, sem a necessidade de buscar dados na RAM.

3º. Nível: A memória principal é disposta na forma de placas de memória que são conectadas a placa mãe em slots próprios obedecendo às especificações da memória. Os processadores Ix possuem um controlador de memória DDR3 de dois canais, suportando memórias de até DDR3-1333

4º. Nível: Memória secundária ou auxiliar, temos aí as memórias conectadas, em geral, através de barramentos (IDE, SATA, USB, SCSI) como as unidades de disco rígido, dispositivos com memória flash como pen drives, entre outros.

Organização de memória do processador Power PC.
http://pds.twi.tudelft.nl/vakken/in101/labcourse/instruction-set/
http://www.cebix.net/downloads/bebox/pem32b.pdf

1º. Nível: Os principais registradores do Power PC são:
32 registradores de uso geral (32 ou 64 bits)
32 registradores de ponto flutuante (64 bits)
8 registradores de condição (32bits)
Registrador de status do ponto-flutuante (32 bits)
Registrador contador (equivalente ao CI, 32 ou 64 bits)
Registrador de ligação (32 ou 64 bits)

2º. Nível: Memória Cache dividida em 2 ubníveis L1 e L2. Processadores Power PC (tomado o G4 como exemplo) possui cache L1 também dividida em 2 partes, cada uma com 32Kbytes. E cache L2 com 128K, 512K ou 1M.

3o. Nível e 4o. Nível são similares à série Ix, diferenciando quanto ao controlador de memória que continua presente no chipset.

OBS : O Power PC G4 é bem anterior ao Intel Ix, as arquiteturas mais atuais que evoluiram do Power PC como o IBM PowerX (Power3,Power6...Power9) passaram a incluir microprogramação em sua arquitetura.