

Arquitectura PowerPC

Visió general, joc d'instruccions i codificació.

2002.12.03

David Geßner
Guillem Cantallops Ramis

Index

Visió General.....	3
Origen.....	3
L'arquitectura PowerPC.....	3
Característiques definides per l'arquitectura PowerPC.....	3
L'arquitectura PowerPC de 64 bits i el subconjunt de 32 bits.....	4
Els nivells de l'arquitectura PowerPC.....	4
Llibertats dins els nivells de l'arquitectura del PowerPC.....	5
Característiques no definides per l'arquitectura PowerPC.....	5
Els registres PowerPC i el model de programació.....	5
Ordenació de bytes.....	6
Organització de dades en la memòria i transmissió de dades.....	7
El model cache del PowerPC.....	7
El model d'excepcions del PowerPC.....	7
El model de gestió de memòria del PowerPC.....	7
Joc d'instruccions i codificació.....	8
Introducció.....	8
Modes d'adreçament.....	8
Instruccions característiques.....	9
Enters.....	9
Punt flotant.....	10
Load/store.....	11
Control de fluxe.....	11
Control de la CPU.....	12
Sincronització de memòria.....	12
Control de memòria.....	13
Control extern.....	13
Conclusió.....	14
Bibliografia.....	15

Visió General

Origen

L'any 1990 hi havia 5 arquitectures RISC que competien al mercat. Era poc probable que totes 5 poguessin tenir èxit. A principis de 1991 les companyies *Motorola*, *IBM* i *Apple Computer* es van juntar per a desenvolupar una arquitectura comú. Com que hagués estat impossible crear una arquitectura completament nova en el temps suficient, es varen decidir a emprar l'arquitectura POWER, implementada per la família RS/6000 d'IBM, com a base. Així sorgí l'arquitectura PowerPC.

L'arquitectura PowerPC

L'arquitectura PowerPC inicialment es va definir com una arquitectura de 32 bits i més tard es va estendre a 64 bits, pensant en la versió de 32 bits com un subconjunt de la de 64 bits. El 32 i el 64 fan referència al tamany dels registres d'enters i als seus registres de suport. En les dues implementacions els registres de punt flotant han estat sempre de 64 bits. En general l'arquitectura defineix el següent:

- El repertori d'instruccions: Especifica les famílies d'instruccions (instruccions del tipus load/store, instruccions d'aritmètica d'enters, instruccions d'aritmètica en punt flotant...), les instruccions específiques i les formes usades per a codificar les instruccions. També especifica els modes d'adreçament usades per accedir a memòria.
- El model de programació: Defineix el conjunt de registres i les convencions per la memòria, incloent els detalls d'ordenació de bit i byte, i les convencions de com es guarden les dades.
- El model de memòria: defineix el tamany de l'espai d'adreces i l'habilitat de configurar pàgines i blocs de memòria.
- El model d'excepcions: defineix el conjunt d'excepcions i les condicions que les poden generar. També defineix els vectors d'excepció (les adreces on comencen les subrutines d'excepció) i el conjunt de registres utilitzats quan es produeix una excepció.
- El model de gestió de memòria: defineix com és subdividida, configurada i protegida la memòria, i com es fa la traducció d'adreces de memòria.
- Model de temporització (*time-keeping*): defineix els mecanismes per determinar l'hora del dia i els recursos i mecanismes d'excepcions relacionats amb el temps.

Característiques definides per l'arquitectura PowerPC

- 32 registres de propòsit general (GPRs, General Purpose Registers) i 32 registres de punt flotant (FPRs, Floating Point Registers). Els GPRs contenen dades per instruccions aritmètiques d'enters i els FPRs contenen dades per instruccions en punt flotant.
- Instruccions per a carregar i guardar dades entre la memòria i els FPRs o GPRs.
- Instruccions de longitud constant per permetre un pipelining (tècnica on es comença a executar una segona instrucció abans de que hagi acabada l'anterior, gràcies a la subdivisió de l'execució de les instruccions en etapes) simplificat d'instruccions i un processament paral·lel d'instruccions.
- Ús no destructiu de registres per instruccions aritmètiques on el segon i el tercer operand especifiquen registres d'origen i el primer operand un

registre destí.

- Un model d'excepcions precís (amb l'opció de tractar excepcions de punt flotant de manera imprecisa). Una interrupció precisa és aquella que permet completar totes les instruccions d'un pipeline que estan per davant de la instrucció que ha fallat, i permet que totes les que estan darrera de la instrucció fallada es puguin reiniciar.
- Suport per nombres en punt flotant que inclouen operacions de punt flotant segons l'estàndar IEEE-754.
- Una definició flexible de l'arquitectura que permet que certes característiques es realitzin a nivell hardware o a nivell software segons les necessitats de la implementació concreta de l'arquitectura.
- Habilitat tant per operacions en punt flotant amb precisió simple com doble.
- Instruccions a nivell d'usuari per a manipular explícitament les dades de les caches sobre el chip. L'arquitectura també defineix instruccions especials per a carregar dades de manera especulativa abans de que faci falta per tal de reduir la latència de la memòria.
- Definició d'un model de memòria que permet accessos a memòria feblement ordenats (*weakly-ordered memory accesses*). Això fa possible reordenar les operacions de bus dinàmicament, incrementant el rendiment general i reduint la latència de la memòria en particular.
- Suport per caches de dades i caches d'instruccions separades i suport per caches unificades.
- Suport per modes d'adreçament big i little-endian.
- Suport tant per implementacions de 32 bits com per implementacions de 64 bits.

L'arquitectura PowerPC de 64 bits i el subconjunt de 32 bits

L'arquitectura PowerPC és de 64 bits amb un subconjunt de 32 bits. Per tant es poden distingir els següents modes d'operació:

- Implementacions de 64 bits en mode 64 bits: tenim un adreçament de 64 bits, tipus enters de 64 bits i instruccions aritmètiques per aquest tipus de dades, a més d'altres característiques per suportar l'adreçament amb 64 bits (per exemple la gestió de memòria és un poc distinta en processadors de 64 i de 32 bits). La CPU es configura per operar en mode 64 bits posant a 1 un bit en el registre d'estat de la màquina (MSR, Machine State Register).
- Processadors que només implementen la part de 32 bits de l'arquitectura PowerPC: tenen adreces efectives de 32 bits i un màxim de 32 bits per dades de tipus enter.
- Implementacions de 64 bits en mode 32 bits: per raons de compatibilitat amb implementacions de 32 bits, les implementacions de 64 bits es poden configurar per a operar en mode 32 bits. En aquest mode les direccions efectives són tractades com adreces de 32 bits i l'aritmètica és de 32 bits (per exemple es produeix overflow quan el resultat supera els 32 bits).

Els nivells de l'arquitectura PowerPC

L'arquitectura PowerPC es defineix en tres nivells que corresponen a tres àmbits de programació:

- UISA (User Instruction Set Architecture): defineix el nivell de l'arquitectura pels programes d'usuari. És a dir, defineix el repertori d'instruccions, els

- registres, els tipus de dades, les excepcions... per al nivell d'usuari.
- VEA (Virtual Environment Architecture): defineix funcionalitats addicionals pel nivell d'usuari que es surten dels típics requeriments dels programes d'usuari. Com per exemple el model de memòria quan més d'un dispositiu pot accedir a la memòria, aspectes del model del cache, instruccions de control del cache...
- OEA (Operating Environment Architecture): defineix els recursos pel nivell supervisor, requerits típicament pel sistema operatiu. El OEA defineix els registres del nivell supervisor, el model de gestió de memòria, els requeriments de sincronització i el model d'excepcions.

Llibertats dins els nivells de l'arquitectura del PowerPC

L'arquitectura PowerPC defineix aquells paràmetres que són necessaris per a assegurar la compatibilitat entre els distints processadors de la família. Però també permet un ampli conjunt d'opcions per implementacions individuals, per exemple:

- Té una sèrie de paràmetres que són opcionals (determinats registres, instruccions...).
- Permet a les implementacions afegir registres de propòsit especial (SPRs, *Special-Purpose Registers*) addicionals. També permet afegir interrupcions i instruccions per requeriments especials (per exemple per processadors dissenyats per a consumir poca energia).
- Els processadors poden implementar qualsevol habilitat o instrucció amb l'assistència de software (per exemple emular) mentre els resultats (a part del rendiment) siguin els mateixos als especificats per l'arquitectura.

Característiques no definides per l'arquitectura PowerPC

La flexibilitat és un aspecte important en el disseny de l'arquitectura PowerPC. És per això que hi ha molts aspectes no definits per l'arquitectura, normalment relacionats amb la implementació hardware. Per exemple no estan definits:

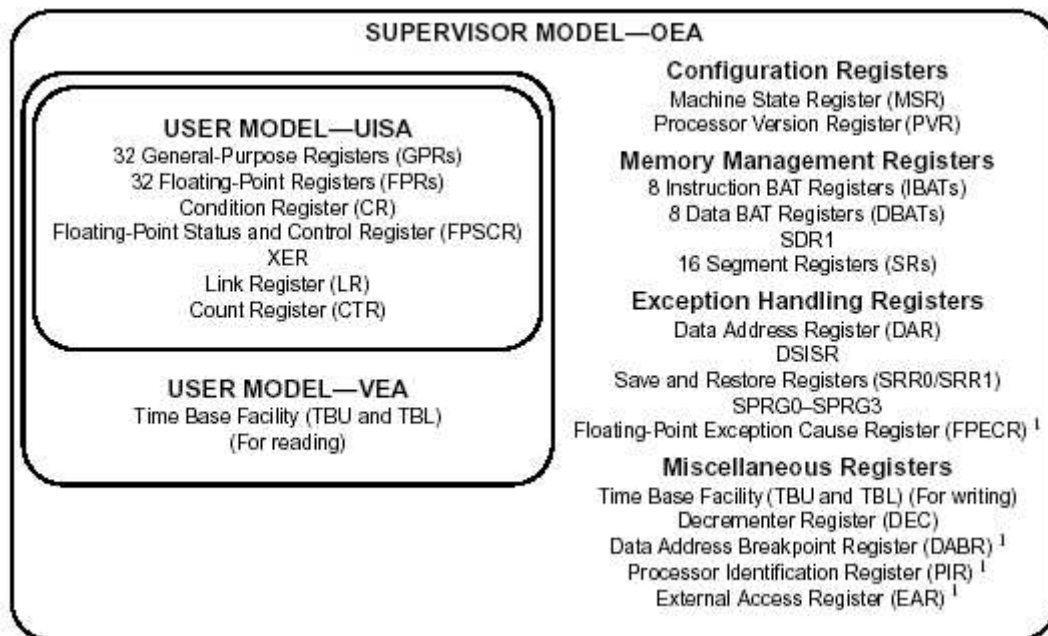
- Els senyals d'interfície amb el bus del sistema.
- El disseny del *cache*.
- El nombre i la naturalesa de les unitats d'execució.
- Quina unitat d'execució és responsable d'executar una instrucció en particular.
- Els detalls del mecanisme de *fetch*.
- El mecanisme de decodificació de les instruccions.

Els registres PowerPC i el model de programació

L'arquitectura *PowerPC* defineix operacions registre a registre per les instruccions de càlcul. Als operands font s'accedeix des dels registres o són aportats directament com valors immediats dins del codi de la instrucció. El format d'instrucció amb tres registres permet tenir un registre destí i dos registres font. Això permet una planificació eficient del codi en processadors altament paral·lels.

Les úniques instruccions que transfereixen dades entre els registres i la memòria són les instruccions *load* i *store*.

Els registres del *PowerPC* es mostren en la següent figura:



Hi ha 32 registres de propòsit general (GPRs) i 32 registres de punt flotant (FPRs). A més hi ha registres de propòsit especial (SPRs) i una sèrie de registres diversos (*miscellaneous*). Cada implementació particular de l'arquitectura pot tenir el seu propi conjunt de registres particulars (registres HID, *Hardware Implementation Dependent*).

Els processadors PowerPC tenen dos nivells de privilegi i que governen l'accés als registres:

- Mode supervisor: usat exclusivament pel sistema operatiu. Els recursos definits per l'OEA només poden ser accedits per software a nivell supervisor.
- Mode usuari: usat pels programes d'aplicació i el sistema operatiu (només els recursos definits per UISA i VEA poden ser accedits per software d'usuari).

Per tant els registres es poden classificar en:

- Registres UISA: poden ser accedits per qualsevol software, tant amb privilegis d'usuari com amb privilegis de supervisor. Aquests registre inclouen els 32 registres de propòsit general i els 32 registre de punt flotant, i altres registres usats per instruccions de bot, d'enters i nombres en punt flotant.
- Registres VEA: el VEA defineix la part del nivell d'usuari del servei de base de temps (*time base facility*), la qual cosa consisteix en dos registres de base de temps de 32 bits. Aquests registres poden ser llegits per programes d'usuari però només poden ser escrits per programes en mode supervisor.
- Registres OEA: els registres de propòsit especial (SPRs) definits pel OEA són usats per operacions del sistema com la gestió de memòria, les excepcions i el manteniment / la gestió del temps (*time-keeping*).

Ordenació de bytes

Per defecte els processadors *PowerPC* son *big-endian*, però també existeix

l'opció de treballar en mode *little-endian*. Això s'aconsegueix mitjançant un bit en el registre d'estat de la màquina (MSR, *Machine Status Register*).

Organització de dades en la memòria i transmissió de dades

Els bytes en la memòria són numerats secuencialment començant per 0. Cada nombre és l'adreça del byte corresponent.

Els operands de la memòria poden ser *bytes*, *half words* (16 bits), *words* (32 bits) o *double words* (64 bits), o, per les instruccions *load/store* string/instruccions múltiples, una seqüència de *bytes* o *words*. L'adreça d'un operand que ocupa múltiples *bytes* és l'adreça del primer dels seus *bytes*, és a dir del *byte* amb la numeració menor. La longitud de l'operand és implícita per a cada instrucció.

Les paraules estan alineades en la memòria.

El model cache del PowerPC

Les parts VEA i OEA de l'arquitectura defineixen aspectes de la implementació de cache pels processadors *PowerPC*. Encara que l'arquitectura *PowerPC* no defineix els aspectes hardware de les implementacions de la cache, sí que estan definides una sèrie d'instruccions com per exemple les operacions per carregar, guardar i buidar continguts de la memòria cache.

El model d'excepcions del PowerPC

El mecanisme d'excepcions (anomenades interrupcions a altres arquitectures) del *PowerPC* permet al processador canviar al mode supervisor quan es reben senyals externes o es produeixen errors. Quan es produeix una excepció la informació sobre l'estat del processador es guardada a determinats registres i el processador comença a executar a partir d'una adreça (vector d'excepció) predeterminada per cada tipus d'excepció.

El model de gestió de memòria del PowerPC

Les especificacions de la unitat de gestió de memòria (MMU, *Memory Management Unit*) estan definides en l'OEA. Les funcions principals de la MMU són traduir les adreces lògiques a adreces físiques per l'accés a memòria i l'accés a entrada/sortida, i de proporcionar una protecció a l'accés de pàgines, blocs o segments.

Els processadors *PowerPC* necessiten una traducció d'adreces per dos tipus de transaccions: accés a instruccions i accés a dades en la memòria (típicament generades per instruccions del tipus *load* i *store*). El mecanisme de traducció es du a terme mitjançant registres de segment i taules de pàgines.

Normalment hi sol haver TLBs (*Translation Lookaside Buffers*) per a guardar les entrades de pàgines usades recentment, però les seves característiques no les especifica l'arquitectura.

Joc d'instruccions i codificació

Introducció

Aquí només tractarem les instruccions de l'arquitectura *PowerPC* limitada a 32 bits (PPC32). S'ha de tenir present que les diferències amb PPC64 seran les mínimes imprescindibles, i en general resultarà fàcil entendre PPC64 coneixent PPC32 i viceversa.

Més enllà dels tres nivells UISA, VEA i OEA, podem classificar les instruccions seguint altres criteris com ara el format utilitzat per a codificar-les. L'arquitectura *PowerPC* defineix moltes instruccions per ser una arquitectura RISC, i algunes son relativament complexes, de manera que per a codificar-les en una longitud fixa el format d'instrucció és bastant complex.

Per exemple el MIPS només té tres formats bàsics (R-Format, I-Format i J-Format), però el *PowerPC* en té cinc (I-Form, B-Form, SC-Form, D-Form i X-Form) que a més internament no son tan senzills com els del MIPS. Especialment complex és l'X-Form del *PowerPC*, l'especificació del qual ocupa una gran taula de quasi 30 línies amb molts de camps petits distribuïts de forma irregular i amb un gran nombre d'excepcions: aquest format s'utilitza per moltíssimes instruccions distintes.

Una forma tal vegada més clara de veure el joc d'instruccions del *PowerPC* consisteix en classificar-les en categories funcionals: enters, punt flotant, *load/store*, control de fluxe, control del processador, sincronització de memòria, control de memòria i control extern. Això indirectament també provoca un agrupament de formats d'instrucció afins, però d'una manera més lògica i amb exemples.

Les instruccions d'enters operen amb paraules (32 bits), mitges paraules i dobles paraules. Les de punt flotant operen amb registres que son de 64 bits en tots els casos, tant en PPC32 com en PPC64. Aquestes instruccions aritmètiques i lògiques no treballen mai directament amb memòria sino amb registres. Per tant, per a utilitzar operands en memòria no queda més remei que utilitzar instruccions tipus *load/store*.

Abans de continuar, és important ressaltar que el *PowerPC* comparteix moltes característiques amb altres *CPUs RISC* com ara el *MIPS*, però també té molts de detalls diferenciadors. Els més importants els veurem tot seguit.

Modes d'adreçament

Per una part, el *PowerPC* té els mateixos modes d'adreçament del *MIPS*:

- **Directe Registre**, a on l'operand és un registre (es codifica l'identificador del registre). Ja que moltes operacions (totes excepte les de *load/store*) son entre registres, aquest mode d'adreçament és molt utilitzat.
- **Indirecte Registre amb Desplaçament**, a on l'operand és a l'adreça que resulta de sumar el contingut d'un registre més un desplaçament, especificat en forma de *constant immediata* (s'especifica explícitament i forma part de l'instrucció codificada).
- **Immediat**, a on l'operand és una constant immediata (també és molt utilitzat).
- **Relatiu al PC**, a on l'operand és a l'adreça que resulta de sumar al PC una constant immediata.

- **Pseudodirecte**, a on l'operand és a l'adreça que resulta de concatenar la part alta del PC amb una constant immediata.

Però el *PowerPC* també té dos modes d'adreçament nous que no té el *MIPS*. Aquí tenim una petita comparació *MIPS* vs. *PowerPC*, a on podem veure que amb el *MIPS* hauriem d'utilitzar dues instruccions per a fer determinades operacions, mentres que amb el *PowerPC* basta una si utilitzam els modes d'adreçament complexos (que no fan que el *PowerPC* deixi d'esser *RISC*, però recorden més a les architectures *CISC*):

- **Indexat**, a on l'operand és a l'adreça que resulta de sumar els dos registres especificats:

```
add $t0, $a0, $s3          vs.      lw $t1, $a0+$s3
lw $t1, 0($t0)
```

- **Amb actualització (update)**, a on l'operand és a l'adreça que resulta de sumar el contingut d'un registre amb una constant immediata, i posteriorment s'actualitza el contingut del registre sumant-li el tamany en bytes de la paraula llegida. Vendria a ser un indirecte registre amb desplaçament i postincrement:

```
lw $t0, 4($s3)           vs.      lwu $t0, 4($s3)
addi $s3, $s3, 4
```

Instruccions característiques

Una vegada més, el *PowerPC* ens sorprèn amb característiques que semblen més típiques de *CISC* que de *RISC*: encara que per la tònica general sigui clarament *RISC*, aquestes instruccions son relativament complexes i especialitzades.

- **Load/store múltiple**: podem llegir/escriure de/a memòria blocs de fins a 32 paraules, utilitzant simultàniament tots els registres de propòsit general. Això ens permet per exemple copiar ràpidament regions de memòria o salvar i restaurar tots els registres d'una manera senzilla i eficient.
- **Bucles tipus `for(i=n;i;i--)`**: tenim un registre comptador de propòsit específic, el qual dona suport a una instrucció especial per a fer bucles amb el nombre d'iteracions predeterminat. Fent una petita comparació *MIPS* vs. *PowerPC* podem veure que amb el *PowerPC* basta una, mentres que amb *MIPS* n'hem d'executar dues.

```
loop: ...                loop: ...
addi $t0, $t0, -1        vs.      bc loop, $ctr!=0
bne $t0, $zero, loop
```

Ara veurem els diferents tipus d'instruccions del *PowerPC*, amb exemples d'algunes instruccions concretes. No hem d'oblidar que el *PowerPC* també té *pseudoinstruccions* com el *MIPS*, però anomenades *mnemònics simplificats*. Essencialment és el mateix, només canvia la nomenclatura.

Enters

El *PowerPC* permet fer operacions aritmètiques, de comparació, lògiques, de rotació i de desplaçament amb els enters.

Com a exemple de grup d'operacions aritmètiques amb enters tenim a la taula següent totes les variants de la suma. Obviament el prefixe *add* significa *sumar*. Els diferents sufixes combinats signifiquen *carrying*, *extended*, *immediate*, *shifted*, *minus one*, *zero*. Sempre que hi ha un camp d'un bit anomenat Rc, a qualsevol instrucció, podem decidir entre actualitzar o no el registre de condició després d'executar-la.

En tots els casos, el registre (aquí sempre enter!) D és el destí de la suma i el registre A és un dels orígens (un dels sumands). L'altre sumand, o bé és el registre B o bé és un valor immediat de 16 bits amb signe (el dibuix està a escala, i ja sabem que en PPC32 totes les instruccions son de 32 bits).

addx	31	D	A	B	OE	266	Rc
addcx	31	D	A	B	OE	10	Rc
addex	31	D	A	B	OE	138	Rc
addi	14	D	A	SIMM			
addic	12	D	A	SIMM			
addic.	13	D	A	SIMM			
addis	15	D	A	SIMM			
addmex	31	D	A	0 0 0 0 0	OE	234	Rc
addzex	31	D	A	0 0 0 0 0	OE	202	Rc

També tenim aquí un exemple d'operacions lògiques bit a bit amb enters, l'*and*. Els sufixes son *with complement*, *immediate* i *shifted*. Els camps van igual que a la suma, només és que l'operand immediat de 16 bits és sense signe (posam UIMM en lloc de SIMM).

andx	31	S	A	B	28	Rc
andcx	31	S	A	B	60	Rc
andi.	28	S	A	UIMM		
andis.	29	S	A	UIMM		

Punt flotant

El PowerPC permet fer operacions aritmètiques, de multiplicació-suma, de redondeig/conversió, de comparació, i de modificació del Registre de Control i Estat amb els nombres de punt flotant.

L'exemple que tenim aquí és d'un tipus especial d'instrucció de punt flotant anomenat de multiplicació-suma, que probablement és molt útil per a fer convolucions entre altres coses. Té quatre operands: el destí és el registre D (tots els registres aquí de punt flotant!) i els registres A, B i C son l'origen. Bàsicament fa $D = (A * C) + B$ a les versions que duen *add* al nom, i $D = (A * C) - B$ a les versions que duen *sub* al nom. Els modificadors son *single precision* (sense modificador per defecte és doble), i *negative* (que li canvia el signe al resultat).

fmadd_x	63	D	A	B	C	29	Rc
fmadds_x	59	D	A	B	C	29	Rc
fmsub_x	63	D	A	B	C	28	Rc
fmsubs_x	59	D	A	B	C	28	Rc
fnmadd_x	63	D	A	B	C	31	Rc
fnmadds_x	59	D	A	B	C	31	Rc

Load/store

El PowerPC pot fer diversos tipus de *load/store*, i son tots molt importants ja que son la única forma de comunicar-se amb la memòria que tenen els programes (la resta d'instruccions son entre registres).

Els tipus de *load/store* del PowerPC son: d'enters, d'enters amb inversió, d'enters múltiples, d'*strings* i de punt flotant.

A continuació tenim un exemple de *load/store* múltiple, de tots els registres o només d'alguns (son les dues úniques instruccions d'aquest tipus). L'adreça efectiva de memòria a partir de la qual fem l'operació es calcula sumant el contingut del registre A amb la constant immediata *d* i el nombre de registres que llegim o escrivim és variable, començam respectivament al registre D o al registre S fins al darrer registre (el 31). Degut a l'alineació de paraules del *PowerPC*, l'adreça efectiva ha d'esser múltiple de quatre (igual que a altres instruccions que treballen amb paraules de memòria, però aquestes son un bon exemple).

lmw ¹	46	D	A	<i>d</i>
stmw ¹	47	S	A	<i>d</i>

Aquí també tenim com a exemple les instruccions de *load/store* d'*strings* (arrays o cadenes de *bytes*, no paraules!). L'adreça efectiva vé donada pel registre A i D / S és el registre a partir del qual escrivim / llegim els bytes individuals. Els modificadors son *immediate* (NB és una constant immediata que diu quants de bytes hem de processar) o *indexed* (el registre B és qui diu quants de bytes hem de processar).

lswi ¹	31	D	A	NB	597	0
lswx ¹	31	D	A	B	533	0
stswi ¹	31	S	A	NB	725	0
stswx ¹	31	S	A	B	661	0

Control de fluxe

Les instruccions de control de fluxe com el seu nom indica son les que controlen el fluxe d'execució dels programes. En el PowerPC tenim les instruccions lògiques que actuen sobre el registre condicional, les instruccions de bot, i els *traps*. Com a exemple aquí tenim les instruccions de bot, amb quatre variants bàsiques (el camp AA indica si l'adreça és absoluta o relativa, i el camp LK indica si s'ha d'actualitzar un registre

de propòsit específic, el LR o *link register*, amb l'adreça de la instrucció que vé després de la instrucció de bot):

- *bx*: Incondicional a l'adreça especificada (absoluta o relativa segons AA).
- *bcx*: Condicional, BI especifica el bit que s'utilitza del registre de condició (CR) i BO acaba de concretar les opcions (botar si és fals, botar si és vertader, manipular el registre comptador, etc.). Es bota a BD que és relativa o absoluta segons AA.
- *bcctrx*: Condicional, respecte del registre comptador (CTR).
- *bclrx*: Condicional, respecte del registre d'enllaç (LR, un registre de propòsit específic per a adreces).

<i>bx</i>	18	LI			AA	LK
<i>bcx</i>	16	BO	BI	BD		AA LK
<i>bcctrx</i>	19	BO	BI	0 0 0 0 0	528	LK
<i>bclrx</i>	19	BO	BI	0 0 0 0 0	16	LK

Control de la CPU

Les instruccions de control del processador requereixen normalment privilegis de supervisor, i habitualment manipulen registres de propòsit específic com ara el MSR (*Machine State Register*), ja siguin registres complets o només parts.

<i>mcrxr</i>	31	crfS	00	0 0 0 0 0	0 0 0 0 0	512	0
<i>mfer</i>	31	D		0 0 0 0 0	0 0 0 0 0	19	0
<i>mfmsr</i> ¹	31	D		0 0 0 0 0	0 0 0 0 0	83	0
<i>mf spr</i> ²	31	D		spr		339	0
<i>mftb</i>	31	D		tpr		371	0
<i>mtrcf</i>	31	S	0	CRM	0	144	0
<i>mtmsr</i> ¹	31	S		0 0 0 0 0	0 0 0 0 0	146	0
<i>mt spr</i> ²	31	D		spr		467	0

Sincronització de memòria

Aquestes instruccions controlen aspectes sobre tot de l'ordre de l'execució de les instruccions, especialment de les d'accés a memòria (*load/store*). Per exemple, *ei eio* significa **enforce in-order execution of i/o**, *isync* només acaba quan totes les instruccions precedents que estaven encara en execució han acabat, etc.

<i>ei eio</i>	31	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	854	0
<i>isync</i>	19	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	150	0
<i>lwarx</i>	31	D	A	B	20	0
<i>stwcx.</i>	31	S	A	B	150	1
<i>sync</i>	31	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	598	0

Control de memòria

Aquestes instruccions controlen essencialment la *caché*, però també permeten manipular el Registre de Segment i el TLB. A la taula que tenim a continuació podem veure les instruccions de control de la *caché*.

El prefixe *dcb* correspon a *data cache block*, i els sufixes corresponen a *allocate*, *flush*, *invalidate*, *store*, *touch*, *touch for store*, i *clear to zero*. Finalment, *icbi* es refereix a la *caché* d'instruccions, que només es pot invalidar (*instruction cache block invalidate*).

En tots els casos, les adreces efectives es calculen sumant els continguts dels registres A i B (els gràfics estan a escala, i tant A com B son de 5 bits la qual cosa els permet a cada un referir-se a un registre concret d'entre els 32 possibles).

<i>dcb</i> ¹	31	00000	A	B	758	0
<i>dcbf</i>	31	00000	A	B	86	0
<i>dcbi</i> ²	31	00000	A	B	470	0
<i>dcbst</i>	31	00000	A	B	54	0
<i>dcbt</i>	31	00000	A	B	278	0
<i>dcbtst</i>	31	00000	A	B	246	0
<i>dcbz</i>	31	00000	A	B	1014	0
<i>icbi</i>	31	00000	A	B	982	0

Control extern

Aquestes instruccions sembla que s'utilitzen per a mapejar adreces per a la comunicació amb dispositius externs, com ara targetes gràfiques. Només n'hi ha dues: *eciwx* per a les entrades i *ecowx* per a les sortides.

<i>eciwx</i>	31	D	A	B	310	0
<i>ecowx</i>	31	S	A	B	438	0

Conclusió

- Per ser una arquitectura *RISC*, el *PowerPC* té un repertori bastant gran d'instruccions, sobre tot si el comparem amb *MIPS*.
- També té algunes instruccions i modes d'adreçament complexos que surten una mica de la filosofia *KISS* (Keep It Simple, Stupid ;-) de *RISC*.
- En qualsevol cas la major part de característiques son típiques de *RISC*: un gran nombre de registres, instruccions majoritàriament senzilles i codificades totes amb la mateixa longitud, operacions entre registres excepte *load/store*, etc.
- *PowerPC* està especificat de forma molt flexible, fins al punt de permetre implementacions de 32 ó 64 bits, amb instruccions opcionals i possibilitat d'afegir-ne de noves.
- Com a exemple proper tenim l'*IBM PowerPC 970*, que probablement acabarà essent la *CPU* dels nous *Macs* d'*Apple*. És un derivat del *POWER4*, de 64 bits, i entre altres coses soporta *AltiVec*, o *Velocity Engine*, que és un conjunt d'unes 160 instruccions de tipus *SIMD*, afegides a les de *PowerPC*.

Bibliografia

Bàsicament hem utilitzat aquests documents en format PDF que es troben a les webs d'*IBM* i de *Motorola*, els principals dissenyadors (i fabricants) de *PowerPC*:

- [http://www-3.ibm.com/chips/techlib/techlib.nsf/techdocs/852569B20050FF778525699600719DF2/\\$file/6xx_pem.pdf](http://www-3.ibm.com/chips/techlib/techlib.nsf/techdocs/852569B20050FF778525699600719DF2/$file/6xx_pem.pdf)
- <ftp://www6.software.ibm.com/software/developer/library/l-ppc.pdf>
- <http://e-www.motorola.com/brdata/PDFDB/docs/MPC7450UM.pdf>
- <http://e-www.motorola.com/brdata/PDFDB/docs/MPCFPE32B.pdf>
- <http://e-www.motorola.com/brdata/PDFDB/docs/MPCPRGREF.pdf>
- [http://www-3.ibm.com/chips/techlib/techlib.nsf/techdocs/A1387A29AC1C2AE087256C5200611780/\\$file/PPC970_MPF2002.pdf](http://www-3.ibm.com/chips/techlib/techlib.nsf/techdocs/A1387A29AC1C2AE087256C5200611780/$file/PPC970_MPF2002.pdf)

També hem utilitzat aquest llibre i les seves extensions Web:

- David A. Patterson, John L. Hennesy. Computer Organization and Design, The Hardware/Software Interface, Second Edition. Morgan Kaufmann, San Francisco, California, 1997.