68020

Pierre Philippe Delacroix

COLLABORATORS

	TITLE : 68020	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Pierre Philippe Delacroix	August 19, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

68020

1.1 Les instructions du processeur 68020

Le 68020 est compatible avec le 68010, ici ne sont détaillées que les
nouvelles instructions ou les instructions ayant subi des modifications.
Voyez les explications sur le 68000 et 68010 pour une decription
complète.

Le micro-processeur 68020 contient une mémoire cache de 256 octets
utilisables, ce qui accélère ses performances en réduisant le nombre
d'accès à la mémoire externe. Le bus est ainsi davantage disponible pour
d'autres processeurs.

Les instructions marquées * sont privilégiées. Elles déclenchent donc
l'exception "Violation de privilège" (n\textdegree{}8) si elles sont exécutées en mode
utilisateur.

Bcc
Branchement conditionnel.

BFCHG
Changement de l'état d'un champ de bits.

BFCLR
Mise à 0 d'un champ de bits.

BFEXTS
Extraction signée d'un champ de bits.

BFEXTU
Extraction non signée d'un champ de bits.

BFFFO
Recherche du premier 1 dans un champ de bits.

BFINS
Insertion d'un champ de bits.

```
                BFSET
                Mise à 1 d'un champ de bits.

                BFTST
                Test de l'état d'un champ de bits.

                BKPT
                Création d'un point d'arrêt.

                BRA
                Branchement inconditionnel.

                BSR
                Branchement à une sous-routine.

                CALLM
                Appel d'un module.

                CAS
                Comparaison et échange

                CAS2
                Comparaison et échange avec 2 opérandes.

                CHK2
                Verification d'un registre avec ses bornes.

                CMPI
                Comparaison avec une donnée immédiate.

                CMP2
                Comparaison d'un registre avec ses bornes.

                cpBcc
                Branchement selon condition de coprocesseur.

                La suite...
                 Information indisponible dans cette démonstration.
 Les codes de conditions
 L'adressage de la mémoire
 Branchements sur Amiga    .
```

1.2 Branchement conditionel

```
Bcc

DESCRIPTION:

Cette instruction réalise un branchement conditionel. Le déplacement peut
être codé sur 32 bits. Tout le reste est identique à l'instruction 68000
correspondante: le branchement à l'étiquette est effectué si la condition
est vraie.

Voir l'avertissement au sujet des branchements relatifs.
```

SYNTAXE:

Bcc Étiquette

TAILLES: octet, mot, mot long.

CC: X N Z V C
    - - - - -

ATTENTION: On ne peut pas utiliser de branchement court sur l'instruction suivante. Il faut pour cela utiliser un branchement mot.

1.3 Test et inversion d'un champ de bits

BFCHG

DESCRIPTION:

Cette instruction permet de tester et d'inverser chaque bit d'un champ. Les codes de conditions sont positionnés selon la valeur du champ.

SYNTAXE:

BFCHG <EA>{offset:largeur}

TAILLE: aucune.

CC: X N Z V C
    - * * 0 0

1.4 Test et mise à 0 d'un champ de bits

BFCLR

DESCRIPTION:

Cette instruction teste le champ de bits et le met à zéro.

SYNTAXE:

BFCLR <EA>{offset:largeur}

TAILLE: aucune.

CC: X N Z V C
    - * * 0 0

1.5 Extraction signée d'un champ de bits

BFEXTS

DESCRIPTION:

Cette instruction permet l'extraction signée d'un champ de bits. Le signe du champ est étendu à 32 bits et sa valeur est chargé dans le registre de données destination.

SYNTAXE:

BFEXTS <EA>{offset:largeur},Dn

TAILLE: aucune.

CC: X N Z V C
    - * * 0 0

1.6 Extraction non signée d'un champ de bits

BFEXTU

DESCRIPTION:

Cette instruction permet l'extraction non signée d'un champ de bits. La valeur du champ est étendue jusqu'au mot long par des zéros et sa valeur est chargé dans le registre de données destination.

SYNTAXE:

BFEXTU <EA>{offset:largeur},Dn

TAILLE: aucune.

CC: X N Z V C
    - * * 0 0

1.7 Recherche du premier 1 dans un champ de bits

BFFFO

DESCRIPTION:

Cette instruction recherche dans le champ de bits le premier bit à 1. L'offset du bit en question est mis dans le registre de données destination. Si aucun bit à 1 n'est trouvé dans le champ, Dn est chargé avec l'offset du champ ajouté à sa largeur.

SYNTAXE:

BFFFO <EA>{offset:largeur},Dn

TAILLE: aucune.

CC: X N Z V C
    - * * 0 0

1.8 Insertion d'un champ de bits

BFINS

DESCRIPTION:

Cette instruction charge le champ de bits indiqué dans l'adresse effective avec les bits de poids faible correspondants du registre de données. La valeur insérée est testée et les codes de conditions sont réglés en conséquence.

SYNTAXE:

BFINS Dn,<EA>{offset:largeur}

TAILLE: aucune.

CC: X N Z V C
    - * * 0 0

1.9 Mise à 1 d'un champ de bits

BFSET

DESCRIPTION:

Cette instruction met à 1 le champ de bits indiqué dans l'adresse effective.

SYNTAXE:

BFSET <EA>{offset:largeur}

TAILLE: aucune.

CC: X N Z V C
    - * * 0 0

1.10 Test d'un champ de bits

BFTST

DESCRIPTION:

Cette instruction permet de tester l'état d'un champ de bits. Les codes de

conditions sont positionnés en conséquence.

SYNTAXE:

BFTST <EA>{offset:largeur}

TAILLE: aucune.

CC: X N Z V C
    - * * 0 0

1.11 Création d'un point d'arrêt

BKPT

Cette instruction sert à signaler la présence d'un point d'arrêt à un débogueur ou un émulateur matériel. Un cycle spécial est activé sur le bus et la donnée immédiate est placée sur les lignes d'adresses A2, A3 et A4. Si ce cycle n'est pas reconnu par un dispositif matériel, une exception "instruction illégale" est générée.

ATTENTION: Cette instruction réagit différemment d'un processeur à l'autre.

SYNTAXE:

BKPT #<Données>   ; Valeurs : 0 à 7.

TAILLE: aucune.

CC: X N Z V C
    - - - - -

1.12 Branchement inconditionnel

BRA

DESCRIPTION:

Cette instruction réalise un branchement inconditionnel. Le déplacement peut être codé sur 32 bits. Tout le reste est identique à l'instruction 68000 correspondante.

Voir l'avertissement au sujet des branchements relatifs.

SYNTAXE:

BRA Étiquette

TAILLES: octet, mot, mot long.

CC: X N Z V C
    - - - - -

ATTENTION: On ne peut pas utiliser de branchement court sur l'instruction suivante. Il faut pour cela utiliser un branchement mot.

1.13 Branchement à une sous-routine

BSR

DESCRIPTION:

Cette instruction réalise un branchement à une sous-routine avec empilement de l'adresse de retour. Le déplacement peut être codé sur 32 bits. Tout le reste est identique à l'instruction 68000 correspondante.

Voir l'avertissement au sujet des branchements relatifs.

SYNTAXE:

BSR Étiquette

TAILLES: octet, mot, mot long.

CC: X N Z V C
    - - - - -

ATTENTION: On ne peut pas utiliser de branchement court sur l'instruction suivante. Il faut pour cela utiliser un branchement mot.

1.14 Appel à un module

CALLM

DESCRIPTION:

Cette instruction assure une subdivision de 256 niveaux d'accès hiérarchisés de l'état utilisateur du microproceseur.

ATTENTION:Cette instruction n'existe que sur 68020, elle n'existe pas sur 68030 et supérieurs!

SYNTAXE:

CALLM #<donnée>,<AE>

TAILLE: aucune.

CC: X N Z V C
    - - - - -

1.15 Comparaison et échange

CAS

DESCRIPTION:

Cette instruction compare l'adresse effective avec le registre de données de comparaisons (Dc). S'ils sont égaux, le contenu du registre de mise à jour (Du) est copié dans l'adresse effective. Dans le cas contraire, Dc est chargé avec le contenu de l'adresse effective.

Cette instruction est particulièrement utile pour les systèmes multi-processeurs, pour la gestions de queues et de piles partagées. Les codes de conditions sont positionnés selon le résultat de la comparaison.

ATTENTION: Cette instruction utilise un cycle de lecture-modification-écriture indissociable. Cela permet de synchroniser plusieurs processeurs. Vous ne devez pas utiliser cette instruction sur Amiga car ce cycle entre en conflit avec les accès DMA de la machine.

SYNTAXE:

CAS Dc,Du,<AE>

TAILLES: octet, mot, mot long.

CC: X N Z V C
    - * * * *

1.16 Comparaison et échange avec 2 opérandes

CAS2

DESCRIPTION:

La première opérande de destination, fournie par l'indirection du registre Rn1, est comparée au registre de données Dc1. En cas d'égalité, la seconde opérande de destination, obtenue par indirection du registre Rn2, est comparée au registre de données Dc2. En cas d'égalité de ces deux derniers, le registre Du1 est copié à l'adresse (Rn1) et Du2 dans (Rn2).

En cas d'inégalité, (Rn1) est copié dans Dc1 et (Rn2) dans Dc2.

ATTENTION: Cette instruction utilise un cycle de lecture-modification-écriture indissociable. Cela permet de synchroniser plusieurs processeurs. Vous ne devez pas utiliser cette instruction sur Amiga car ce cycle entre en conflit avec les accès DMA de la machine.

Cette instruction n'existe que sur les processeurs 68020, 68030 et 68040. Elle n'existe pas sur 68060.

SYNTAXE:

CAS2 dc1:Dc2,Du1:Du2,(Rn1):(Rn2)

TAILLES: octet, mot, mot long.

```
CC: X N Z V C
    - * * * *
```

1.17 Vérification d'un registre avec ses bornes

CHK2

DESCRIPTION:

Le registre Rn est comparé avec les valeurs indiquées en mémoire, à l'adresse donnée par l'opérande <AE>. La première valeur à cette adresse est considérée comme la borne inférieure et la seconde comme borne supérieure. Si le registre n'est pas entre ces deux valeurs, l'exception de vecteur 6 est déclenchée.

Le test est effectué autant pour des limites signées que non signées. Si Rn est un registre d'adresses, les bornes sont étendues jusqu'au mot long avant la comparaison.

ATTENTION: Cette instruction n'existe que sur les processeurs 68020, 68030 et 68040. Elle n'existe pas sur 68060, mais sera facilement émulée par un programme approprié.

SYNTAXE:

CHK2 <AE>,Rn

TAILLES: octet, mot, mot long.

```
CC: X N Z V C
    - ? * ? *
```

1.18 Comparaison avec une donnée immédiate

CMPI

DESCRIPTION:

Cette instruction ôte la donnée immédiate de l'opérande destination. Les codes de conditions sont alors positionnés selon le résultat. La destination n'est pas modifiée. L'adresse effective peut maintenant utiliser l'adressage relatif au PC. Tout le reste est identique à l'instruction 68000 correspondante.

SYNTAXE:

CMPI #<donnée>,<AE>

TAILLES: octet, mot, mot long.

CC: X N Z V C

```
    - * * * *
```

1.19 Comparaison d'un registre avec ses bornes

```
CMP2

DESCRIPTION:

Le contenu du registre Rn est comparé avec les deux valeurs adjacentes
stockées à l'adresse indiquée par l'opérande <AE>. La première est
considérée comme la borne inférieure et la seconde comme la borne
supérieure. Si Rn est un registre d'adresses, les bornes sont étendues
jusqu'au mot long avant la comparaison.

Le bit Z du registre des codes de conditions est mis à 1 si Rn est égal à l'une
de ses bornes et à 0 dans le cas contraire. La retenue C est à 1 si la valeur
de Rn est hors de ces bornes, à 0 dans le cas contraire.

ATTENTION: Cette instruction n'existe que sur les processeurs 68020, 68030
et 68040. Elle n'existe pas sur 68060, mais sera facilement émulée par un
programme approprié.

SYNTAXE:

CMP2 <AE>,Rn

TAILLES: octet, mot, mot long.

CC: X N Z V C
    - ? * ? *
```

1.20 Préfixe cp

```
ATTENTION

Les mnémoniques au format cpXXX n'existent pas!

Ces pseudos-instructions signifient en fait que le processeur supporte
les instructions correspondantes des FPU (68881 et 68882) ou MMU (68851).

Dans ces conditions,cpXXX doit être compris comme :
- FXXX si l'on s'adresse aux coprocesseurs mathématiques 68881 et 68882.
- PXXX si l'on s'adresse au coprocesseur d'unité de gestion de la
  mémoire 68851.
```

1.21 Branchement selon une condition de coprocesseur

```
                cpBcc

DESCRIPTION:
```

Cette
pseudo-instruction
teste la condition du coprocesseur indiquée et réalise un branchement à l'étiquette indiquée si elle est vraie. La condition dépend du processeur sélectionné.

Voir l'avertissement au sujet des branchements relatifs.

SYNTAXE:

cpBcc Étiquette

TAILLES: mot, mot long.

CC: X N Z V C
- - - - -

1.22 Décrémentation et branchement selon condition de coprocesseur

La suite ...
Information indisponible dans cette démonstration.

1.23 Exécute une instruction d'un coprocesseur

La suite ...
Information indisponible dans cette démonstration.

1.24 Restauration de l'état d'un coprocesseur

La suite ...
Information indisponible dans cette démonstration.

1.25 Sauvegarde de l'état d'un coprocesseur

La suite ...
Information indisponible dans cette démonstration.

1.26 Positionnement selon une condition de coprocesseur

La suite ...
Information indisponible dans cette démonstration.

1.27 Déclenchement d'une exception selon une condition de coprocesseur

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.28 Division signée

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.29 Division non signée

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.30 Extension de signe

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.31 Création de liens avec la pile

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.32 Transfert d'un registre de contrôle

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.33 Multiplication signée

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.34 Multiplication non signée

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.35 Compression de binaire codé décimal

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.36 Retour de module

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.37 Test d'une opérande

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.38 Piégeage du processeur selon les codes de conditions

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.39 Décompression de binaire codé décimal

```
La suite ...
 Information indisponible dans cette démonstration.
```

1.40 Ceci n'est qu'une démonstration

```
Vous trouverez toutes ces informations et bien d'autres choses dans
la version française exclusive des Editions A.D.F.I.

Editions A.D.F.I.
résidence les cottages
83 rue André Theuriet
F-63000 Clermont Ferrand

Téléphone : 3304+ 73.93.77.31.
```