Mapping Info Structure

Dsx-vm needs to have a way to communicate with the operating system to specify how to map code, data and taks on the architecture. To this end, an xml data structure has been defined, and is presented below. This xml file is produced by Dsx-vm and read by the operating system and used in particular for page tables generation and thread mapping on processors.

TODO: hypothesis: clustered_architecture, etc.

Document XML Structure

In order to simplify the document type definition, all element attributes are mandatory.

The mapping_info element

The mapping file is constituted of a single `mapping_info` element, containing the following attributes:

- `name`: the name of the mapping described
- `signature`: *what is it?* (Mohamed?)
- `clusters`: the number of clusters used in the mapping. It must correspond to the number of clusters in the targeted architecture
- `vpsaces`: the number of vspaces defined. This number is typically equal to the number of applications (requirement?)

The `mapping_info` element contains the following elements:

- a `clusterset` element, describing the targeted hardware
- a `globalset` element, describing the global segments, i.e. segments replicated in all virtual spaces
- a `vspaceset` element, describing the mapping for all virtual spaces

The clusterset (cluster set) element

This element contains no attribute, and contains the following elements:

- 1 to *n* `cluster` element(s)

The cluster element

This element contains the following attribute:

- `index`: the cluster index, or *id*. It is unique for each cluster and must range from 0 to *n - 1*. The cluster index make the correspondance with the cluster id in the architecture.

The cluster element contains the following elements:

- 0 to *n* `pseg` element(s)
- 0 to *n* `proc` element(s). The number of `proc` elements must be equal to the number of processors in this cluster in the targeted architecture (doit être le même dans chaque cluster ?).
- 0 to *n* `periph` element(s)

The pseg (physical segment) element

This element contains the following attributes:

- `name`: name of the segment. The name of the segment is not related to any segment name in the architecture. It is used to associate virtual segments on physical segments inside the mapping info file.
- `base`: base of the physical segment
- `type`: type of the segment. It can be one of `RAM`, `ROM`, `PERI` (influence ??)
- `length`: size of the segment (utilisé pour des vérifications ?)

The proc (processor) element

This element contains the following attribute:

- `index`: the processor index, or *id*. It is unique for each processeur and must range from 0 to *n - 1* inside the cluster (vrai ? ou id global ?). The index corresponds to the processor id in the architecture

The proc element contains the following element:

- 1 `irq` element

The irq (interrupt request) element

This element contains the following attributes:

- `type`: The type of interrupt request. Currently can only be `HARD` (?)
- `icuid`: the identifier of ..?
- `isr`: the interrupt sub-routine...what?
- `channel`: the channel of the... (toujours égal à icuid ?)

The periph (peripheral) element

This element contains the following attributes:

- `type`: type of the peripheral. It can be one of `TTY`, `TIM` (timer), ?
- `psegname`: name of the physical segment in which the peripheral is mapped. This name must be one of the pseg elements' name. (pourquoi nom du pseg et pas nom du vseg ?)
- `channels`: the number of channels in the peripheral (one channel usually corresponds to one processor; an exception is for ttys where one channel is used for the operating system)

The globalset (global set) element

This element contains no attribute, and contains the following elements:

- 0 to *n* `vseg` element(s)

The vseg (virtual segment) element

This element contains the following attributes:

- `name`: name of the virtual segment (utilisé quelque part ?)
- `vbase`: base address of the virtual segment in the final binary. Depending on the virtual segment, it can be chosen arbitrarily or not (détailler)
- `clusterid`: *pourquoi est-ce que c'est nécessaire ?*
- `psegname`: name of the physical segment in which this virtual segment must be mapped. This name must be one of the pseg elements' name.
- `mode`: specific properties of the segment. The value is of the form `[C_][X_][W_][U_]` e.g. `C_W_`. Each character indicates if the property is selected (letter) or not (underscore). The properties are the following:
    - `C`: the segment is cached
    - `X`: the segment is executable
    - `W`: the segment is writable
    - `U`: the segment is accessible is user mode
- `ident`: is 1 if the segment is an identity segment, 0 otherwise. An identity virtual segment is a virtual segment whose base address is the same as the one of the physical segment it is mapped on. *Pourquoi a-t-on besoin de ça ? de toutes façons toutes les infos sont répliquées...*

The vseg element contains the following elements:

- 1 to *n* `vobj` element(s)

The vobj (virtual object) element

This element contains the following attributes:

- `name`: the name of the virtual object. *utilisé ?*
- `type`: type of the virtual object. It can be one of `ELF`, `PERI`, `BUFFER`, `MWMR`, `PTAB` (influence ?)
- `length`: size of the virtual object in bytes. *utilisé pour quoi ?*
- `align`: logarithm in base 2 of the alignemnt required for this segment. For example, a value of 13 means that the segment must be aligned on 0x2000
- `binpath`: path to the file to load in this segment, if any. This path can be relative or absolute.

The vspaceset (virtual space set) element

The vspaceset element has no attribute and contains the following elements:

- 0 to *n* `vspace` element(s)

The vspace (virtual space) element

This element contains the following attributes:

- `name`: name of the virtual space set (utilisé ?)
- `startname`: ??

The vspace element contains the following elements:

- 0 to *n* `vseg` element(s)
- 0 to *n* `task` element(s)

The task element

This element contains the following attributes:

- `name`: name of the task. It must be the name of the main function of the task.
- `clusterid`: cluster id in the targeted architecture of the processor on which to execute the task.
- `proclocid`: processor local id on which to execute the task.
- `stackname`: name of the vobj on which to place the task's stack. It must the name of one of the vseg element.
- `usetty`: is 1 if the task has to write its output on a tty, 0 otherwise.
- `startid`: (j'ai oublié ce que c'est)